UNISYS

V Series MCP/VS System Software

Operations Guide

Volume 3: System Utilities

Relative to Release
Level Mark 2.0

Priced Item

September 1987
Distribution Code SD
Printed in U S America
5026255

UNISYS

V Series
MCP/VS
System Software

Operations Guide

Volume 3: System Utilities

Copyright © 1987 Unisys Corporation
All Rights Reserved
Unisys is a trademark of Unisys Corporation

Relative to Release
Level Mark 2.0

Priced Item

September 1987
Distribution Code SD
Printed in U S America
5026255

The names, places, and/or events used in this publication are not intended to correspond to any individual, group, or association existing, living, or otherwise. Any similarity or likeness of the names, places, and/or events with the names of any individual living or otherwise, or that of any group or association is purely coincidental and unintentional.

NO WARRANTIES OF ANY NATURE ARE EXTENDED BY THE DOCUMENT. Any product and related material disclosed herein are only furnished pursuant and subject to the terms and conditions of a duly executed Program Product License or Agreement to purchase or lease equipment. The only warranties made by Unisys, if any, with respect to the products described in this document are set forth in such License or Agreement. Unisys cannot accept any financial or other responsibility that may be the result of your use of the information in this document or software material, including direct, indirect, special or consequential damages.

You should be very careful to ensure that the use of this information and/or software material complies with the laws, rules, and regulations of the jurisdictions with respect to which it is used.

The information contained herein is subject to change without notice. Revisions may be issued to advise of such changes and/or additions.

Comments or suggestions regarding this document should be submitted on a Field Communication Form (FCF) with the CLASS specified as 2 (S.W.:System Software), and the Type specified as 1 (F.T.R.), and the product specified as the 7-digit form number of the manual (for example, 5026255).

Publication Change Notice (PCN)

UNISYS

Date	Form — PCN Number
September 1988	**5026255-002**

Title

V Series MCP/VS Systems Software Operations Guide Volume 3: System Utilities

Description

This PCN updates the document to reflect the 2.2 release (September, 1988).

Replace these pages	Add these pages
ix through x	—
xiii through xiv	—
xxi through xxii	—
2-1 through 2-2	—
4-3 through 4-4	—
5-3 through 5-8	—
5-11 through 5-14	—
5-17 through 5-18	—
10-1 through 10-2	—
13-1 through 13-2	—
17-1 through 17-8	—
18-1 through 18-9	—
	28-1 through 28-3

Copyright © 1988 Unisys Corporation

Distribution Code SD

Printed in the United States

Publication Change Notice (PCN)

UNISYS

Date	Form — PCN Number
May 1988	**5026255-001**

Title

V Series MCP/VS Systems Software Operations Guide Volume 3: System Utilities

Description

This PCN updates the document to reflect the 2.1 release (May, 1988).

Replace these pages	Add these pages
v through xvii	—
xxi	—
4-1 through 4-5	—
6-1 through 6-2	—
7-11 through 7-17	7-18-A
7-27 through 7-29	—
10-1 through 10-5	—
11-1 through 11-5	—
14-1 through 14-11	—
22-1 through 22-9	—
23-23	—
	24-1 through 27-2

Copyright © 1988 Unisys Corporation

Distribution Code SD

Printed in the United States

AFFECTED PAGES

Page	Issue	Page	Issue
Title	Original	24-1 through 2424-5	PCN-001
v through xiii	PCN-001	24-6	Blank
xiv	Blank	25-1 through 25-33	PCN-001
xv through xvii	Original	25-34	Blank
xviii	Blank	26-1	PCN-001
xix through xx	Original	27-1 through 27-2	Original
xxi through xxii	PCN-00	1 through 6	Original
1-1 through 1-2	Original		
2-1 through 2-14	Original		
3-1 through 3-5	Original		
3-6	Blank		
4-1 through 4-6	PCN-001		
5-1 through 5-27	Original		
5-27	Blank		
6-1 through 6-2	PCN-001		
7-1 through 7-10	Original		
7-11 through 7-18-A	PCN-001		
7-18-B	Blank		
7-19 through 7-26	Original		
7-27 through 7-30	PCN-001		
7-31 through 7-36	Original		
8-1 through 8-11	Original		
8-11	Blank		
9-1 through 9-2	Original		
10-1 through 10-5	PCN-001		
11-1 through 11-6	PCN-001		
11-7 through 11-18	Original		
12-1	Original		
12-2	Blank		
13-1 through 13-5	Original		
13-6	Blank		
14-1 through 14-12	PCN-001		
15-1 through 15-3	Original		
15-4	Blank		
16-1 through 16-15	Original		
16-16	Blank		
17-1 through 17-7	Original		
17-8	Blank		
18-1 through 18-9	Original		
18-10	Blank		
19-1 through 19-16	Original		
20-1 through 20-3	Original		
20-4	Blank		
21-1 through 21-11	Original		
21-12	Blank		
22-1 through 22-9	PCN-001		
22-10	Blank		
23-1 through 23-22	Original		
23-23 through 23-24	PCN-001		
23-25 through 23-31	Original		
23-32	Blank		

AFFECTED PAGES

Page	Issue	Page	Issue
Title	Original	17-1 through 17-8	PCN-002
v through vii	PCN-001	18-1 through 18-9	PCN-002
ix through x	PCN-002	18-10	Blank
xi through xii	PCN-001	19-1 through 19-16	Original
xiii through xiv	PCN-002	20-1 through 20-3	Original
xv through xvii	PCN-001	20-4	Blank
xviii	Blank	21-1 through 21-11	Original
xix through xx	Original	21-12	Blank
xxi through xxii	PCN-002	22-1 through 22-9	PCN-001
1-1 through 1-2	Original	22-10	Blank
2-1 through 2-2	PCN-002	23-1 through 23-22	Original
2-3 through 2-14	Original	23-23 through 23-24	PCN-001
3-1 through 3-5	Original	23-25 through 23-31	Original
3-6	Blank	23-32	Blank
4-1 through 4-2	PCN-001	24-1 through 24-5	PCN-001
4-3 through 4-4	PCN-002	24-6	Blank
4-5 through 4-6	PCN-001	25-1 through 25-33	PCN-001
5-1 through 5-2	Original	25-34	Blank
5-3 through 5-8	PCN-002	26-1	PCN-001
5-9 through 5-10	Original	27-1 through 27-2	Original
5-11 through 5-14	PCN-002	28-1 through 28-3	PCN-002
5-15 through 5-16	Original	1 through 6	Original
5-17 through 5-18	PCN-002		
5-19 through 5-27	Original		
5-28	Blank		
6-1 through 6-2	PCN-001		
7-1 through 7-10	Original		
7-11 through 7-18-A	PCN-001		
7-18-B	Blank		
7-19 through 7-26	Original		
7-27 through 7-30	PCN-001		
7-31 through 7-36	Original		
8-1 through 8-11	Original		
8-11	Blank		
9-1 through 9-2	Original		
10-1 through 10-2	PCN-002		
10-3 through 10-5	PCN-001		
11-1 through 11-6	PCN-001		
11-7 through 11-18	Original		
12-1	Original		
12-2	Blank		
13-1 through 13-2	PCN-002		
13-3 through 13-5	Original		
13-6	Blank		
14-1 through 14-12	PCN-001		
15-1 through 15-3	Original		
15-4	Blank		
16-1 through 16-15	Original		
16-16	Blank		

TABLE OF CONTENTS

Section	Title	Page
	About This Document	xix
	Purpose	xix
	Scope	xix
	Audience	xix
	Prerequisites	xix
	Results	xix
	How To Use This Document	xix
	Organization	xix
	Related Product Information	xxii
1	B974LD Intrinsic Program	1-1
	Introduction	1-1
	Network Initialization	1-1
	Recovery of Messages	1-2
	Additional Tasks	1-2
	System and Program Dump Files	1-2
	B 974 System Maintenance Logs	1-2
	Printer Backup Files	1-2
	NDL Compilation	1-2
2	File Transfer Utility (SYSTEM/COPY)	2-1
	Program Flow	2-1
	Security	2-2
	Implementation Maximums	2-2
	BNA File Transfer	2-3
	Error Messages	2-3
	Warning Messages	2-5
	Library Maintenance Messages	2-5
	I/O Error Handling and Messages	2-6
	Additional Error Possibilities	2-7
	Recovery Options	2-7
	Reliability Handling and Messages	2-8
	Missing File Handling and Messages	2-9
	Missing Diskpacks	2-9
	Missing Disk and Diskpack Files	2-9
	Missing Tape Files	2-9
	Duplicate Disk and Diskpack Files	2-10
	Non-library Tape Handling and Messages	2-10
	LOADMP/PACKUP Tape Handling and Messages	2-10
	ICTAPE Format	2-10
	Reading An ICTAPE	2-11
	TAPE Format	2-11
	Reading a TAPE	2-13

TABLE OF CONTENTS (Continued)

Section	Title	Page
	Programmatically Differentiating TAPE, ICTAPE, LOADMP and PACKUP Tape Formats	2-13
	Visually Differentiating TAPE, ICTAPE, LOADMP and PACKUP Tape Formats	2-13
3	Data Communications Processor (DCP)	3-1
	System Requirements	3-1
	Network Initialization	3-1
	MCP Interface for B 874 or B 974	3-2
	DLP record	3-2
	DCP UNIT Record	3-2
	LIMIT DCPQUE Record	3-2
	LIMIT DCPBUF Record	3-2
	Command SO with DCP	3-2
	Command LH	3-2
	BUFFER Control Instruction	3-2
4	DLP Utilities (DLPXNO AND DLPXCO)	4-1
	Executing DLPXNO	4-2
	Executing DLPXCO	4-3
	Error Messages	4-5
5	File Conversion Utility Program (DMPALL)	5-1
	Using DMPALL to List Files	5-2
	Syntax for Listing Files with DMPALL	5-2
	Syntax elements for Listing Files with DMPALL	5-5
	Using DMPALL To Convert Files	5-10
	Overview of Syntax for Converting Files with DMPALL	5-11
	Detailed Syntax for Converting Files with DMPALL	5-12
	Detailed Media Conversion Parameters	5-14
	Miscellaneous DMPALL Functions	5-20
	Listing LOADMP and PACKUP Library Files	5-20
	Listing Object Code Files from Disk	5-20
	Listing A Series Printer Backup Tapes	5-20
	Displaying Tape Directories	5-20
	Listing LOADMP and PACKUP Tape Directories	5-21
	ZIP Mechanism	5-21
	Starting DMPALL with the EXECUTE Command	5-22
	How to Start DMPALL with the EXECUTE Command	5-22
	VALUE 0 = 100000	5-22
	VALUE 0 = 000001	5-23
	VALUE 0 = 001000	5-23
	When DMPALL Encounters a Parity Error	5-24
	DMPALL Examples	5-25
6	Memory Dump Utility (DMPMEM)	6-1
	Executing DMPMEM	6-1

TABLE OF CONTENTS (Continued)

Section	Title	Page
7	Diskpack Utility Program (DISPKV)	7-1
	Execution Syntax	7-2
	INITIALIZE Command	7-10
	CONFIGURE Command	7-13
	REPORT Command	7-15
	SINGLE Command	7-16
	RELOCATE Command	7-18
	RENAME Command	7-19
	RECONFIGURE Command	7-20
	RECONFIGUREL Command	7-21
	LABEL Command	7-22
	659IVR Command	7-24
	Response To ODT Input Messages	7-26
	Displaying Status Information	7-26
	Initializaing MD4, MD8, 680, and 682 Diskpack	7-27
	Diskpack and Controller Styles Supported	7-27
	File Equate Information	7-30
	Input Examples	7-30
	Results of Input	7-31
	Card Input Rules	7-31
	DISPKV Messages	7-32
	Informational Messages	7-32
	Operator Prompts	7-33
	Error Messages	7-34
	Initializing Diskpacks To 100-Byte Mode	7-36
8	Pseudo Reader Load Control Program (LDCNTL)	8-1
	LDCNTL System Commands and Options	8-1
	LDCNTL MCP Control Instructions	8-2
	LDCNTL Keyboard Input Messages	8-2
	When Not to Use Pseudo Card Files	8-3
	Creating a Psuedo Card File	8-3
	File Names	8-3
	Requirements for a Control Deck	8-4
	Initiating Load Control	8-4
	MCP Capabilities in LDCNTL	8-5
	Operating Procedures	8-5
	Setting Up	8-5
	DELETING PSEUDO CARD FILES	8-6
	RECOVERY OF PSEUDO CARD FILES	8-6
	Examples of Typical Pseudo Card Files	8-7
	Files and Records	8-9
	File Assignment	8-9
	Reader of File Deallocation	8-10

TABLE OF CONTENTS (Continued)

Section	Title	Page
	LDCNTL - Punch Backup to Psuedo Card File Conversion	8-10
9	Host Load Intrinsic Program (LDHOST)	9-1
	Host Load for B 974 DCPs	9-1
	Host Load For B 874 DCPs	9-1
	Automatic Operation	9-1
	How to Load The LOS Into An Intellibent Laser Printer	9-2
10	Off-line Firmware Loader Program (LOADFW)	10-1
	Executing LOADFW	10-1
	Normal Messages	10-3
	Error Messages	10-4
11	Floppy Disk Copy Utility Program (MDCOPV)	11-1
	Supported Hardware	11-1
	Format Information	11-1
	Operating Instructions	11-2
	Executing MDCOPV and Setting its Switch Settings	11-2
	Copy Functions	11-2
	Copying Files *to* a Floppy Disk	11-3
	Copying Files *from* a Floppy Disk	11-3
	Duplicating Floppy Disks	11-3
	Copy Function Command Syntax	11-4
	Duplicating Floppy Disks With The Copy Function	11-6
	File and Floppy Disk Analyze Functions	11-6
	Copy Function For Floppy Disk Image Files	11-9
	Conversion Functions	11-11
	Examples	11-12
	File Equate Information	11-13
	MDCOPV Messages	11-13
	Informational Messages	11-13
	Input Prompts	11-14
	Error Messages	11-15
12	MCP Merge Intrinsic Program (MERG:)	12-1
	Functional Description	12-1
13	DCP Utility Program (NIFMRG)	13-1
	How To Operate NIFMRG	13-1
	Parameter Syntax	13-2
	DCP Statement	13-2
	MERGED Statement	13-3
	END Statement	13-3
	QUIT Statement	13-4
	Statements Longer Than One Line	13-4
	Errors	13-5
	Listing	13-5
14	PBDPRN Utility Program (PBDPRN)	14-1

TABLE OF CONTENTS (Continued)

Section	Title	Page
	Examples With Explanations	14-5
	Executing PBDPRN	14-5
	Auto Printing	14-5
	PBDPRN Syntax Errors	14-6
	AX Errors	14-7
	Messages Related to the Image Page Printer and Line Printers	14-8
	Messages Related to the Image Page Printer	14-9
	Image Page Printer Error Recovery Messages	14-11
15	Diskpack Copy Utility (PKCOPY)	15-1
	Operating Instructions	15-1
	PKCOPY Messages	15-2
16	Shared Systems and Devices (SHARED)	16-1
	Single-System Shared Environment	16-4
	Multisystem Shared System Environment	16-4
	Types of Shared Systems	16-5
	Shared System	16-5
	Single-System Shared	16-5
	MultiSystem Shared	16-5
	Shared Disk Only System	16-5
	Shared Diskpack Only System	16-6
	Shared Disk And DiskPack System	16-7
	Definitions of Shared File I/O Operations And Other Related Terms	16-8
	Components Of A Shared System	16-9
	Configuration And Initialization Of Shared SYstems	16-10
	Single System Shared	16-10
	Multiple-System Shared	16-11
	File 1	16-12
	File 2	16-13
	Stacked Deck	16-14
17	Sort Intrinsic Program (SORT.)	17-1
	Input	17-1
	Sorting	17-2
	Output	17-2
	Memory Requirements	17-2
	Disk Requirements	17-3
	Special Provisions For Sorting Variable-Length Records	17-3
	Site-specific Sort Parameters	17-4
	Error Messages	17-5
18	Tape/Disk Sort Intrinsic Program (SORT:)	18-1
	Memory Requirements	18-1
	Key specifications	18-1
	Input And Output Requirements	18-2
	Input Restrictions	18-2

TABLE OF CONTENTS (Continued)

Section	Title	Page
	Input And Output Assumptions	18-3
	Rules For Sequence Of Output Files	18-3
	Tape Sorting	18-3
	Tape Sort Operation: A short narrative description	18-3
	Operating Considerations	18-3
	Disk Or Diskpack Sorting	18-4
	Disk Sort Media	18-4
	Default Media	18-4
	Work File Requirements	18-4
	Disk Sort Operation: A short narrative description	18-4
	Operating Considerations	18-4
	Default Media Override	18-4
	Virtual Collating Sequence	18-6
	Translate Option	18-7
	Translate Tables	18-7
	Error And Warning Messages	18-7
	Non-Fatal Errors	18-7
	Recovery	18-7
	Fatal Errors - Potentially Correctable	18-7
	Fatal Errors - Non-Correctable	18-8
19	Generalized Sort Utility Program (SRTUTL)	19-1
	SYNTAX	19-1
	FILE Statement	19-2
	MULTIFILE Statement	19-5
	KEY Statement	19-5
	SORT Statement	19-7
	ADDROUT Statement	19-7
	IDENT Statement	19-7
	COMPARE Statement	19-8
	MEMORY Statement	19-9
	PARITY Statement	19-9
	RECORDS Statement	19-10
	Execution Instructions	19-10
	Cardless Executions	19-11
	Label Equation	19-13
	Dollar Options	19-14
20	Automatic System Recovery Facility (SYSUP)	20-1
	SYSUP Programming Considerations	20-1
	Example SYSUP Programs	20-2
21	Uniline DOP Utility (UNLODV)	21-1
	Firmware File USP2BH	21-2
	Firmware File UST2BH	21-3
	MCP Configuration Records	21-3

TABLE OF CONTENTS (Continued)

Section	Title	Page
	UNLODV Program Flow	21-4
	Executing UNLODV	21-4
	Data Communications Parameters	21-4
	UNLODV Commands	21-10
	Firmware File Format	21-11
22	MCP Memory Dump Analysis Program (DMPANL)	22-1
	Requirements For Dump Analysis	22-1
	Starting DMPANL under MCP/VS 2.n	22-2
	DMPANL Command Syntax	22-2
23	QWIK Disk	23-1
	Performance Improvement Guidelines	23-3
	QWIK Disk Peripheral Performance	23-3
	General System Performance	23-3
	Code File Overlays	23-3
	Sort Performance	23-3
	DMSII and ISAM Performance	23-3
	Transaction System Performance	23-4
	Applying QWIK Disk To Your System	23-4
	System Analysis - An Overview	23-4
	System Analysis	23-5
	Step 1 - Examine Processor Utilization, Find out MCP Idle Time	23-6
	Step 2 - How Much Time Is Spent Waiting MCP and Program Overlays	23-7
	Step 3 - Memory Utilization, Examine MCP and Program Overlays	23-8
	Step 4 - Peripheral Activity, Examine I/O Wait TIME	23-10
	Step 5 - I/O Counts to Disk and Diskpack	23-11
	Step 6 - Prepare for General File I/O Analysis	23-12
	Step 7 - Examine I/O Volume to MCP Files	23-12
	Step 8 - Examine I/O Volume to Classes of Files	23-12
	Step 9 - Examine I/O Volume to Individual Files	23-13
	Step 10 - Channel/Subsystem/ID Analysis	23-14
	Dynamic Profile versus Static Profile	23-14
	Sample Environments	23-15
	Sample Development Environment	23-15
	Sample Production Environment	23-15
	Sample Diskpack Environment	23-15
	Measurement Techniques	23-15
	Using FLAME	23-15
	Files	23-17
	General File Selection Criteria	23-17
	File Size	23-17
	File Access	23-18
	File Volatility	23-18
	File Activity	23-18

TABLE OF CONTENTS (Continued)

Section	Title	Page
	File Selection Suggestions	23-18
	Compiler	23-19
	File Sizing	23-19
	How to Place Files in QWIK Disk	23-19
	QWIK Disk Performance Expectations	23-19
	Sort	23-19
	File Size	23-19
	How to Place Files on QWIK Disk	23-19
	QWIK Disk Performance Expectations	23-20
	DMSII	23-20
	File Size	23-20
	How to Place Files on QWIK Disk	23-20
	QWIK Disk Performance Expectations	23-20
	ISAM	23-21
	File Sizing	23-21
	How to Place Files on QWIK Disk	23-21
	QWIK Disk Performance Expectations	23-21
	GEMCOS	23-22
	File Sizing	23-22
	How to Place File on QWIK Disk	23-22
	QWIK Disk Performance Expectations	23-22
	MCP Configuration Records for QWIK Disk	23-23
	Putting the MCP on QWIK Disk	23-23
	Use QWKMEM and QWIKPOOL Options	23-25
	Shared Systems	23-25
	Operations	23-25
	Maintenance Processor Commands	23-25
	Caution on Maintenance Test Commands	23-26
	Powering On the System	23-26
	Haltload	23-26
	Putting Files on QWIK Disk	23-26
	Using SYSTEM/COPY	23-27
	Loading and Unloading Files From QWIK Disk	23-27
	Firmware	23-27
	Programming Considerations	23-27
	Creating Files	23-27
	Programmatic File Creation on QWIK Disk	23-27
	QWIK Disk as Default Disk	23-28
	Work File Subsystem Default	23-28
	File Equation to QWIK Disk	23-28
	Hardware Type	23-29
	Random and Sequential I/O	23-30
	Code File Overlays	23-30

TABLE OF CONTENTS (Continued)

Section	Title	Page
	A Deviation from Standard Disk Operation	23-30
	Error Conditions	23-30
	Result Descriptors	23-30
	Single Bit Errors	23-30
24	Communications Processor Firware Loading Program (CPLOAD)	24-1
	Initiating CPLOAD	24-1
	LH Command	24-1
	FE*IBL Label (CP 3680 Only)	24-2
	VCS Initiation	24-2
	Error Messages	24-3
25	DEBUG Facility	25-1
	Initiating a DEBUG Session	25-1
	MCP Initiation	25-1
	Utility Commands	25-1
	The DEBUG Command	25-2
	The Interactive DEBUG Command (ID)	25-5
	The Enter DEBUG Command (ED)	25-6
	The Query DEBUG Command (QD)	25-7
	The User Interface Menus	25-9
	The State of a Debugged Task	25-9
	The Main Menu	25-11
	The Status Menu	25-14
	The Trace Menu	25-16
	The Breakpoint Menu	25-18
	The State Menu	25-21
	DEBUG Session Examples	25-23
	Stop User Program	25-23
	Using the PEEK and POKE Functions	25-26
	Using the Trace Function	25-28
	DEBUG Session Errors	25-30
	Error Messages Related to DEBUG Commands	25-33
26	Memory Dump File Copy Utility (DMPCPY)	26-1
	Executing DMPCPY	26-1
27	Object Program Copy Utility (PCOPY)	27-1
	General	27-1
	Operating Instructions	27-1

TABLE OF CONTENTS (Continued)

Section	Title	Page
28	Translation File Generator Program	27-1
	Operating Instructions	28-1
	MAKTRN Parameter Records	28-1
	MAKTRN Options	28-2
	IDNT Option	28-2
	ALFA Option	28-2
	NUMR Option	28-2
	SEQN Option	28-3

LIST OF ILLUSTRATIONS

Figure	Title	Page
Figure 2-1.	Library Maintenance Message	2-5
Figure 2-2.	I/O Error Message Format	2-7
Figure 2-3.	Block Error Message	2-8
Figure 5-1.	DMPALL File Listing Syntax	5-3
Figure 5-1 (Continued).	DMPALL File Listing Syntax	5-4
Figure 5-2.	DMPALL File Conversion Syntax—Overview	5-11
Figure 5-3.	DMPALL File Conversion Syntax—Detailed	5-12
Figure 5-3 (Continued).	DMPALL File Conversion Syntax—Detailed	5-13
Figure 7-1.	DISPKV Syntax	7-2
Figure 7-2.	DISPKV General Syntax	7-3
Figure 7-3.	DISPKV INITIALIZE Syntax	7-10
Figure 7-4.	DISPKV CONFIGURE Syntax	7-13
Figure 7-5.	DISPKV REPORT Syntax	7-15
Figure 7-6.	DISPKV SINGLE Syntax.	7-16
Figure 7-7.	DISPKV RELOCATE Syntax	7-18
Figure 7-8.	DISPKV RENAME Syntax	7-19
Figure 7-9.	DISPKV RECONFIGURE Syntax	7-20
Figure 7-10.	DISPKV RECONFIGUREL Syntax	7-21
Figure 7-11.	DISPKV LABEL Syntax	7-22
Figure 7-12.	DISPKV 659IVR Syntax	7-24
Figure 7-13.	Sample DISPKV Execute Deck	7-32
Figure 8-1.	Typical Pseudo Card File	8-7
Figure 8-2.	Execute Program	8-8
Figure 11-1.	MDCOPV Floppy Disk Copy Syntax	11-4
Figure 11-1a.	MDCOPV Floppy Disk Duplication Syntax	11-6
Figure 11-2.	MDCOPV Flexible Disk Analyze Syntax	11-7
Figure 11-3.	MDCOPV Floppy Disk Image File Syntax	11-10
Figure 11-3.	MDCOPV File Conversion Syntax	11-11
Figure 13-1.	DCP Statement	13-2
Figure 13-2.	Merged Statement	13-3
Figure 13-3.	END Statement	13-4
Figure 13-4.	QUIT Statement	13-4
Figure 14-1.	PRINT Syntax	14-1
Figure 15-1.	PKCOPY Syntax	15-1
Figure 16-1.	Concurrent File Access	16-3
Figure 16-2.	Shared Diskpack Only System	16-6
Figure 16-3.	Shared Disk and Diskpack System	16-7
Figure 19-1.	SRTUTL Program Statements.	19-1
Figure 19-2.	FILE Statement	19-3
Figure 19-3.	MULTIFILE Statement	19-5
Figure 19-4.	KEY Statement	19-5
Figure 19-5.	SORT Statement	19-7
Figure 19-6.	ADDROUT Statement	19-7

LIST OF ILLUSTRATIONS (Cont)

Figure	Title	Page
Figure 19-7.	INDENT Statement	19-8
Figure 19-8.	COMPARE Statement	19-8
Figure 19-9.	MEMORY Statement	19-9
Figure 19-10.	PARITY Statement	19-9
Figure 19-11.	Records Statement	19-10
Figure 19-12.	SRTUTL Execution Card Deck	19-11
Figure 19-13.	SRTUTL Execution Without Cards	19-11
Figure 22-1.	PM Command Syntax for MCP Dumps	22-3
Figure 22-2.	DMPANL Command Parameters	22-3
Figure 22-3.	DMPANL Table Selection Options EXTTBL	22-4
Figure 22-4.	MCP Tables Available in DMPANL Listing	22-5
Figure 22-5.	DMPANL Raw Memory Selection Syntax	22-8
Figure 22-6.	DMPANL Task Selection Syntax	22-8
Figure 22-7.	DMPANL Task Selection Parameters	22-8
Figure 22-8.	DMPANL Task Memory Selection Syntax	22-9
Figure 23-1.	Example Memory Partition Between the MCP and QWIK Disk.	23-2
Figure 23-2.	Comparison of MCP Idle Time to MCP and User Program Execution Time	23-5
Figure 23-3.	Factors that Contribute to MCP Idle Time	23-6
Figure 23-4.	QWIK Disk I/O Performance For Programs Not Using QWIKPOOL	23-7
Figure 23-5.	Find Out What Percentage of Time the MCP Spends Waiting I/O	23-9
Figure 23-6.	Analyze I/O traffic to Disk, Pack, and other Media	23-10
Figure 23-7.	Disk I/O Analysis of a System Waiting I/O to 10 Different Types of Files	23-11
Figure 23-8.	Underconfigured and Lopsided Peripheral Configurations	23-13
Figure 23-9.	Blocksize and Ti me Factors for Completing I/Os to QWIK Disk.	23-29
Figure 25-1.	DEBUG Command Syntax.	25-2
Figure 25-2.	Syntax for Parameter List	25-3
Figure 25-3.	ID Command Syntax	25-5
Figure 25-4.	ED Command Syntax	25-6
Figure 25-5.	QD Command Syntax	25-7
Figure 25-6.	User Interface Menus	25-8
Figure 25-7.	Status Line — Example	25-10
Figure 25-8.	Main Menu	25-11
Figure 25-9.	Status Menu.	25-14
Figure 25-10.	Trace Menu	25-16
Figure 25-11.	Breakpoint Menu	25-18
Figure 25-12	State Menu. 25-21	25-21
Figure 25-13.	Main Menu — Breakpoint Example	25-23
Figure 25-14.	Breakpoint Menu — Address Breakpoint Example	25-24
Figure 25-15.	Main Menu — Example	25-25
Figure 25-16.	Main Menu — PEEK Input Example.	25-26
Figure 25-17.	Main Menu — PEEK Display Example	25-27
Figure 25-18.	Main Menu — POKE Example	25-28
Figure 25-19.	Trace Menu — Example	25-29

LIST OF TABLES

Table	Title	Page
Table 3-1.	B 874 Link Errors	3-3
Table 3-2.	B 974 Link Errors -	3-4
Table 3-3.	Firmware Interface Errors Detected by MCP	3-5
Table 7-1.	DISPKV Parameter Keywords and Default Values	7-4
Table 7-3.	Available Disk and Diskpack Styles	7-28
Table 7-4.	DISPKV Commands and Qualified Disk Types	7-29
Table 9-1	LDHOST-MCP Interfaces	9-2
Table 11-1	File Names for MDCOPV	11-13
Table 16-1.	Shared I/O Operations	16-4
Table 19-1.	Valid INSERT Values for Cardless Execution	19-11
Table 19-2.	Meanings of INSERT Values for Cardless Execution	19-12
Table 19-3.	SRTUTL Files and Format	19-12
Table 19-4.	SRTUTL File Names	19-13

ABOUT THIS DOCUMENT

PURPOSE

This manual documents various utilities available for use with the Unisys MCP/VS operating system. These utilities perform such tasks as copying and moving files, initializing disks and diskpacks, establishing data communications and many other functions. This manual is written as a companion text to the *V Series MCP/VS System Software Operations Guides, Volumes 2, 3, and 4.*

SCOPE

This manual documents what each utility it is used for and how they are operated. It includes commands used with the various utilities, and messages that they display. In many cases, examples of operator input are provided. In some cases, technical details of program interaction or file layout are included.

AUDIENCE

The target audience for this manual is an experienced user of Unisys V Series equipment.

PREREQUISITES

Anyone using this manual should be familiar with the Master Control Program for V Series MCP/VS, and the information included in the other operations guides.

RESULTS

You will be able to use those system utilities that are relevant to your programming and operating environment.

HOW TO USE THIS DOCUMENT

This manual is designed as a companion text to the other operations guides, and can be used with them to operate the system.

ORGANIZATION

This document consists of the following parts, each of which discusses a separate utility program. The sections appear in alphabetic order according to each utility's name.

SECTION 1: B974LD INTRINSIC PROGRAM

The B974LD utility is used with the B 974 Data Communications Processor.

SECTION 2: FILE TRANSFER UTILITY (SYSTEM/COPY)

The SYSTEM/COPY utility copies and converts files from one file storage medium to another, or on the same medium. It copies and moves files to and from disk, diskpack, and tape, and also lists what files reside on various media.

SECTION 3: DATA COMMUNICATIONS PROCESSOR (DCP)

This section discusses in general terms the use of a B 874 or B 974 Data Communication Processor (DCP) to reduce the workload on the V Series system involved in data communications. It lists related MCP configuration records and commands, as well as link error codes and firmware interface error codes for the B 874 and B 974 Data Communication Processors.

SECTION 4: DLP UTILITIES (DLPXNO AND DLPXCO)

A data link processor (DLP) is an interface between the V Series processor and a peripheral device. The DLPXNO and DLPXCO utilities dump and clear data link processors that are not functioning due to irrecoverable I/O errors.

SECTION 5: FILE CONVERSION UTILITY PROGRAM (DMPALL)

The DMPALL utility is a flexible file handling utility. It lists, copies and converts disk, diskpack, and tape files. It can move files from one medium to another. Unlike the SYSTEM/COPY utility, it can deal with files that were not created by a Unisys V Series system, and that have unusual blocking factors, no tape label, or other unusual characteristics. It can list all or part of disk, diskpack, tape, and card files, including object code disk files. It can produce listings in hexadecimal code, and can produce a translation into characters.

SECTION 6: MEMORY DUMP UTILITY (DMPMEM)

The DMPMEM utility is a floppy-disk based utility that dumps MCP memory to tape. It is used to produce memory dumps in abnormal situations that cannot be handled by the MCP intrinsic Fault Handler.

SECTION 7: DISKPACK UTILITY PROGRAM (DISPKV)

The DISPKV utility initializes and purges LAK disk (100-byte-per-sector) and diskpack (180-byte-per-sector) media. It can verify the contents of a disk or diskpack, relocate sectors on a disk or diskpack, change selected fields in a diskpack label, and list the addresses of sectors on a disk or diskpack. It is useful in cases of media failure.

SECTION 8: PSEUDOREADER LOADCONTROL PROGRAM (LDCNTL)

The LDCNTL utility creates pseudo card files from other types of files. These pseudo card files can be used to control the system.

SECTION 9: HOST LOAD INTRINSIC PROGRAM (LDHOST)

The LDHOST utility downloads firmware (microcode) from the V Series processor to devices that are interfaces to peripheral devices. LDHOST can download firmware to a programmable controller, a data communication processor (DCP), or a Uniline device.

SECTION 10: OFF-LINE FIRMWARE LOADER PROGRAM (LOADFW)

The LOADFW utility downloads firmware (microcode) files from the V Series processor to certain devices that interface between V Series processors and peripheral devices or between multiple V Series processors. These devices are the diskpack controller, the shared system processor (SSP), and the Uniline data link processor (Uniline DLP).

SECTION 11: FLOPPY DISK COPY UTILITY PROGRAM (MDCOPY)

The MDCOPY utility copies files to and from floppy disks, and performs floppy disk maintenance functions. This utility recognizes the Industry Compatible Minidisk (ICMD) format and the Micro-Minidisk Unit (MMDU) format.

SECTION 12: MCP MERGE INTRINSIC PROGRAM (MERG:)

The MERG: utility provides file merging functions to utilities written in V Series ANSI-84 COBOL.

SECTION 13: DCP UTILITY PROGRAM (NIFMRG)

The NIFMRG utility merges together files called Network Information Files. These files contain information used for data communications through the B 874 data communications processor, the B 974 data communications processor, the Offline Reader-Sorter (ORS-DLP), and the TELCOM DLP. If more than one type of these devices are to be used at once, the NIF files must be merged by the NIFMRG utility.

SECTION 14: PBDPRN Utility Program (PBDPRN)

The PBDPRN utility prints out printer backup files that have been prepared by other utilities.

SECTION 15: DISKPACK COPY UTILITY (PKCOPY)

The PKCOPY utility copies the data from one diskpack to another.

SECTION 16: SHARED SYSTEMS AND DEVICES (SHARED)

This section discusses the ability of multiple V Series processors to access the same files on the same media, using a shared system processor (SSP).

SECTION 17: SORT INTRINSIC PROGRAM (SORT.)

The SORT. utility provides rapid sorting facilities to programs written in ANSI-68 COBOL and BPL.

SECTION 18: TAPE/DISK SORT INTRINSIC PROGRAM (SORT:)

The SORT: utility is provides rapid sorting facilities to programs written in ANSI-74 COBOL and to the SRTUTL utility.

SECTION 19: GENERALIZED SORT UTILITY PROGRAM (SRTUTL)

The SRTUTL utility is a standalone interface to the SORT. utility. It provides sorting facilities to programs written in RPG and to operators. It is designed to sort indexed sequential, relative, or sequential files according to keys. It also generates either an ADDROUT or sequential file.

SECTION 20: AUTOMATIC SYSTEM RECOVERY FACILITY (SYSUP)

The SYSUP utility helps to automatically recreate the operating environment when a system failure occurs.

SECTION 21: UNILINE DCP UTILITY (UNLODV)

A Uniline DLP is one kind of interface device between a V Series system and a peripheral device. The UNLODV utility helps prepare such a device for use.

SECTION 22: MCP MEMORY DUMP ANALYSIS PROGRAM (DMPANL)

The DMPANL utility produces a formatted analysis of a memory dump of the MCP.

SECTION 23: QWIK DISK

QWIK DISK allows you to use part of system memory as "RAM disk" for rapid-access storage of files.

SECTION 24: COMMUNICATIONS PROCESSOR FIRMWARE LOADING PROGRAM (CPLOAD)

CPLOAD loads firware to communications processors that are connected to the V Series Communication System (VCS).

SECTION 25: DEBUG FACILITY

The Interactive DEBUG facility allows debugging activities (modification of memory, breakpoints, and so forth) for user programs and, optionally, for the operating system.

SECTION 26: MEMORY DUMP FILE COPY UTILITY (DMPCPY)

DMPCPY copies a memory dump file from one magnetic tape to another.

SECTION 27: OBJECT PROGRAM COPY UTILITY (PCOPY)

PCOPY copies object code files from 100-byte disk to 180-byte diskpack and vice versa.

SECTION 28: TRANSLATION FILE GENERATOR PROGRAM (MAKTRN)

MAKTRN creates translation tables that can be used with the sort intrinsic programs (SORT. and SORT:).

RELATED PRODUCT INFORMATION

This manual is designed as a companion text to the following manuals.

- *V Series MCP/VS System Software Operations Guide, Volume 1: Installation Utilities*
- *V Series MCP/VS System Software Operations Guide, Volume 2: System Commands*
- *V Series MCP/VS System Software Operations Guide, Volume 4: MCP Output Messages*

SECTION 1
B974LD INTRINSIC PROGRAM

Introduction

The B974LD program on the host communicates with the HOST LOAD program on the B 974 Data Communications Processor (DCP). The program transfers files, executes Network Definition Language (NDL) compiles, and initializes and activates the communications network.

The *B 974 DCP Installation and Operations Guide* provides procedures for initiating the B974LD program.

This section gives an overview of the B974LD program features.

The host initiates the B974LD program in the following situations:

- When you set the MCP DCP option with the command SO DCP or the system sets the MCP DCP option because of a USE DCP record in the system configuration file.
- When you use the MCP command LH to reactivate the B 974.
- When the host finishes precompiling a Network Definition Language (NDL) source file, and the compilation process needs to be completed on the B 974.
- When a message from the B 974 is to be displayed on the host Operator Display Terminal (ODT).

The B974LD program communicates with only one B 974 at a time. Therefore, hosts with multiple B 974s may have multiple copies of the B974LD program running simultaneously. The program performs the following functions:

- Transfers various files, including printer backup listings, B 974 firmware code files, B 974 system memory dumps, and B 974 resident program dumps to the host.
- Transfers the B 974 system maintenance logs and printer backup files generated by B974MCS trace and other utilities to the host.
- Transfers B 974 NDL source code files, firmware code files, and host Message Control System (MCS) information files from the host to the B 974.
- Initiates and monitors NDL compiles on the B 974 from the host.
- Displays information from the B 974 on the host ODT.
- Activates the data communications network.

Network Initialization

When you use the commands LH, LH WARM, or SO DCP to execute the B974LD program, or if there is a USE DCP record in the MCP configuration file, the MCP attempts to initialize and activate the data communications network.

If you use the command LH, the B974LD program transfers the firmware code file from the host to the B 974. If you use the command LH WARM or SO DCP, the B974LD program transfers the firmware code file

from the host to the B 974 only if the firmware code file on the host is not the same as the firmware code file on the B 974. If the firmware code file is the same on both the host and the B 974, the B974LD program does not transfer the firmware code file.

Recovery of Messages

If the host fails, then all messages queued on the host are lost.

If a firmware code file is transferred to the B 974 when you initialize it, all messages queued on that DCP are lost. If the file is not transferred, the messages are retained. If you want to retain the messages, use the command LH WARM or SO DCP. Please note, you will lose the messages if the system transfers a firmware file, regardless of the command you use. If you want to discard the messages, use the command LH. For more information on when the B974LD program transfers the firmware code file, refer to the *B 974 DCP Installation and Operations Guide*.

Whenever you execute an MCS that was previously active, all messages queued in the B 974 during the host failure will be sent to the MCS.

Additional Tasks

After network initialization or NDL compilation, the B974LD program performs the following tasks.

System and Program Dump Files

The B974LD program uploads system memory dumps or B 974 resident program dumps to the host.

B 974 System Maintenance Logs

The B974LD program analyzes maintenance log files generated by the B 974 and uploads the listings to the host.

Printer Backup Files

The B974LD program uploads printer backup files from the B 974 disk to the host. The files may have been generated by an MCS issuing a trace function header.

NDL Compilation

The B 974 NDL compilation process has two stages, one on the host and one on the B 974. When the stage on the host is completed with no errors, the B974LD program transfers an intermediate code file to the B 974. The B 974 then completes the compilation, which results in a firmware code file.

If the compilation on the B 974 contains no errors, the B974LD program transfers a copy of this completed firmware code file to the host for safekeeping. For more information about this process, refer to the *B 974 DCP Installation and Operations Guide*.

SECTION 2
FILE TRANSFER UTILITY (SYSTEM/COPY)

SYSTEM/COPY is a bound intrinsic that performs file transfer and directory functions for the V Series MCP. It is invoked by the commands COPY, MOVE, COPYADD, MERGE and DIR, which are documented in *V Series MCP/VS System Software Operations Guide, Volume 2: System Commands.* SYSTEM/COPY performs the following functions:

- Copies files from one or more sources at a time.
- Merges files from one or more source tapes and from disk and diskpack to one or more tapes.
- Copies files to one or more destinations at a time.
- Changes the names of files as they are being transferred.
- Copies files from tape to their original volumes.
- Permits flexibility in job scheduling and AFTER executions because simultaneous executions of COPY can utilize a unique multi-program ID.
- Stores files in requested sequence on all output tape volumes.
- Produces two tape formats: ICTAPE (interchange format) and TAPE.
- Recognizes and reads automatically a variety of tape formats: LOADMP (type 5), PACKUP (type 6), TAPE and ICTAPE.
- Permits the use of large, multi-level file names on tapes.
- Copies from any reel of a multi-reel tape set since each reel includes a directory of the files on it and all subsequent reels.
- Supports 38000 BPI magnetic cartridge tapes (MTC).
- Supports 9-track tapes (NRZ, PE and GCR).
- Lists directories for disk, diskpack and tape. See the MCP command DIR for details.

Program Flow

This section provides a high-level program flow description for the file transfer utility (SYSTEM/COPY). SYSTEM/COPY is executed directly by the operating system (MCP/VS) and receives the required syntax through a pass file.

The first step in the program flow is an analysis of the syntax received from the pass file. If any warnings or error messages are encountered, they will be emitted. If no fatal errors are found, the program will then acquire memory to accommodate the necessary number of file structures and buffers. If there is insufficient memory to accommodate the memory size requested, the program may be marked as "PR STOPPED" (priority stopped).

Once appropriate memory has been acquired, file expansion occurs. At this stage of the program flow, group specifiers are expanded to their respective single file names. Keep in mind that this expansion is performed only for those source volumes that require it.

Tape source volumes are also searched for both group specifiers and single files if there are any destination tape volumes. In situations where destination tape volumes are not involved, source tape file expansion is postponed until that volume is actually needed for transfer. This technique is employed to provide better tape drive utilization.

Once complete file lists for all sources are obtained, and when a tape directory has been generated for any destination tape volumes that are found, SYSTEM/COPY is then ready to perform the required tasks.

Processing begins with the first source file. The required functions are performed using the file list associated with that source. This process continues for each source volume (and associated file list).

Each file transfer proceeds first by finding and opening the file on the current source volume. Next, if the COPYADD function is being performed, each destination volume is checked for the presence of the destination file name (which may differ from the source file name when the AS capability is employed). All destination files are then opened.

Each file is transferred with consideration given to minimizing I/O's, and providing effective buffering. As a general rule, mixing destination types (for example, copying a disk file to tape as well as to diskpack) will increase overhead. Most transfers occur in a single pass of the source file, unless disk or diskpack files are transferred to mixed destination media (for example, copying a disk file to tape and to diskpack). In these cases, the source file is first transferred to the "like" media, then to the "unlike" media.

After all transferring is completed, each file is closed according to its transfer type. That is, COPY's and COPYADD's release the source file, while MOVE's remove the source file provided no errors have occurred and a compare is not pending. All destination files that have successfully completed transferring are then closed in a manner that removes any duplicates.

If requested, SYSTEM/COPY will then perform the compare function. Only those files that successfully completed transfer will be eligible for comparison. If MOVE was used as the transfer function, the source file will be removed after a successful comparison is performed.

Finally, if a summary was requested, SYSTEM/COPY generates the ODT portion of the summary, listing only those files that have completed all phases of the transfer successfully. The summary also shows the number of files transferred with and without errors. A printed summary is also generated. It includes the ODT displayed information, and additional information on errors and related operator responses.

Security

Any security attributes associated with a given source file will be copied to all destination files. Similarly, tape files will carry their attributes with them when they are transferred to disk or diskpack.

If a file located on a source tape has corrupt security information, SYSTEM/COPY will assign PUBLIC/IO as its security use and type value when this file is transferred to any destination volumes.

Implementation Maximums

Listed below are the various maximum values associated with the current release of SYSTEM/COPY. The characters "mm" represent COPY's mix number, and the character "p" represents the processor number.

Syntax pass file (mm0p0 - contains the input syntax)

40 segs/area, 20 areas = 800 lines

Directory pass file (mm0p0 - contains directory information)

90 segs/area, 20 areas = 28,784 file ID's

Work file (Wmm0p0 - contains volume and file information)

1000 segs/area, 100 areas = 100,000 records

Record requirements vary with the number of volumes and files you specify, along with the number of file identifiers into which group specifiers are expanded. The maximum number of specified files for each volume is 999,999; likewise, the maximum number of expanded files for each group specifier is also 999,999.

Tape directory file (Tmm0p0 - generated tape directory)

1500 segs/area, 100 areas = 150,000 records.

File maximum depends on file name sizes.

Print file (SYSTEM/COPY)

Unlimited capacity if printer is used; backup capacity is governed by MCP limits.

Destination volumes.

No more than 10 destination volumes may be specified. The number of source volumes is not limited.

BNA File Transfer

SYSTEM/COPY performs file transfer for BNA facilities through the use of the HOSTNAME volume attribute. There are several limitations and requirements necessary to implement the current BNA:

The COPYADD function is not supported.

Transfers involving tapes are not supported.

SYSTEM/COPY will attempt to optimize the transfer by increasing the buffers for each file. When all HOSTNAMEs specified are recognized as the local host, the network and associated transfer scheme will not be used.

Error Messages

Below is a list and a brief explanation of the error messages SYSTEM/COPY can output. When errors are produced, COPY goes to an abnormal end of job.

Error messages will always be displayed on the ODT, regardless of the system or requested options. This means that any library maintenance messages that indicate an error has occurred will have been preceded by an appropriate error message.

In general, SYSTEM/COPY attempts to pinpoint the syntax errors by displaying the text line in question and pointing to the word it was looking at when it found the error.

BUFFERSIZE HAD 3600 DIGIT MINIMUM AND 99000 DIGIT MAXIMUM

The BUFFERSIZE option has been used, and the value specified is not in the required range. BUFFERSIZE has 3600 digit minimum and 99000 digit maximum.

COPYADD FUNCTION ILLEGAL FOR BNA APPLICATION

The COPYADD function is not legal when a HOSTNAME attribute is present.

EXPECTED FILE LIST

A file ID or list of file ID's was expected.

EXPECTED KIND ATTRIBUTE (DISK, PACK, TAPE, ICTAPE, OLDTAPE)

A non-default volume identifier was found without a KIND attribute.

EXPECTED OPTION OR VALID OPTION ABBREVIATION (COMPARE, SUMMARY, ODTSUMMARY, PRNSUMMARY, MPID, MANDATORY, SYNTAX, CHECK, BUFFERSIZE, SPECIFY, ALLOCATED)

An invalid option was found. SPECIFY appears in this message only if the keyword MERGE was used.

EXPECTED VERB (COPY, COPYADD, MOVE, COMPARE, DIR, MERGE)

An invalid verb was found.

EXPECTED "TO"

The beginning of an output specification was expected.

EXPECTED "TRUE" OR "FALSE"

A Boolean SET option argument was illegal.

EXPECTED "("

A left parenthesis was expected.

EXPECTED ")" OR ","

A right parenthesis or comma was expected.

EXPECTED ","

A comma was expected.

INCOMPATIBLE FILE IDS FOR VOLUME KIND

An attempt was made to use a large (greater than 6 characters) or multilevel file ID on disk or diskpack media.

INVALID ATTRIBUTE

An invalid or unsupported file or volume attribute was found.

INVALID ATTRIBUTE ARGUMENT

The value for this file or volume attribute is illegal.

INVALID FILE IDENTIFIER

A file identifier must be less than or equal to 6 characters in length, and begin with an alpha or acceptable special character.

INVALID OPTION ARGUMENT

The specification for this option is illegal.

INVALID TOKEN - EXPECTED "FROM", "TO", OR ","

Another file or source specification or destination specification was expected.

INVALID USE OF "ALLOCATED" OPTION

The ALLOCATED option has been specified, and a source that was specified has also been specified as a destination. This is invalid.

INVALID USE OF "ORIGIN" AS A VOLUME KIND.

Origin was used either as a source kind or as a destination kind when no tape sources were supplied.

INVALID VOLUME IDENTIFIER

A volume identifier must be less than or equal to 6 characters and follow the same rules as a file identifier.

MISSING CLOSE PARENTHESIS

A matching parenthesis was not found for a beginning parenthesis.

MISSING CLOSE QUOTE

A closing quote was not found for an opening quote.

NUMBER OF DESTINATION VOLUMES EXCEEDS MAXIMUM

The current limit of 10 output destinations has been exceeded.

"ON" EXPECTED

Invalid syntax for the DIR command was found. Specification of the volume must be preceded with ON.

TAPE HARDWARE ILLEGAL FOR BNA APPLICATION

TAPE volumes are not supported across a BNA network.

Warning Messages

DATE OPTION IGNORED - NOT IMPLEMENTED

The DATE SET option is not implemented and its specification is ignored.

SAVEFACTOR ATTRIBUTE TOO BIG - 999 SUBSTITUTED

The maximum save factor allowed is 999 days.

UNSUPPORTED ATTRIBUTE IGNORED: attribute

The specified attribute is not currently supported.

Library Maintenance Messages

SYSTEM/COPY generates a library maintenance message for each file it attempts to copy. These messages are displayed on the ODT if the USE COPY option is set.

If the SUMMARY option is used, a listing of all files copied is displayed on the ODT, and a print file listing containing the library maintenance messages is also produced.

The form these messaged is illustrated in Figure 2-1 below:

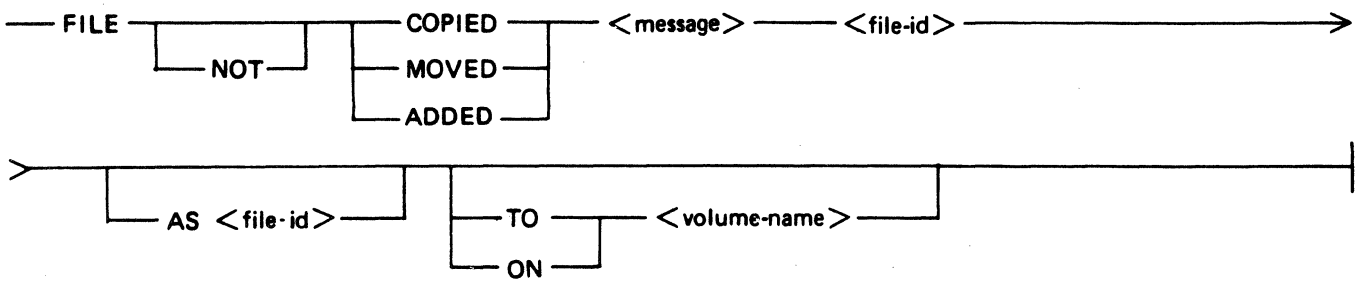


Figure 2-1. Library Maintenance Message

The TO and FROM volume names occur when the ORIGIN option for destination volumes is used.

A brief description of each of the error messages is presented below.

DESTINATION ERROR

An error has occurred on the destination volume. In this circumstance, COPY goes to an abnormal end of job.

FILE ALREADY PRESENT

This occurs when COPYADD is used. A file with a name matching the file to be copied already exists at the destination.

FILE NOT COPIED - UPDATED BY MERGE: <file-id>

The file <file-id> was not copied because of a MERGE.

FILE NOT COMPARED - UPDATED BY MERGE: <file-id>

The file <file-id> was not copied, and therefore not compared, because of a merge.

FILE NOT PRESENT

The requested file was missing from the source volume. If the MANDATORY options is set, COPY goes to an abnormal end of job.

INTERNAL ATTRIBUTE ERROR

An irrecoverable error has occurred in the validation of the file header at file open time. In this circumstance, COPY goes to an abnormal end of job.

NOT ON REQUESTED VOLUME INDEX

The file was not on the tape reel requested. Since the tape directory contains all files residing on this and any subsequent reels of a volume set, COPY will use this message to indicate any files that did not begin on the requested reel. When this occurs, COPY goes to an abnormal end of job.

OK

The stated function occurred without error.

VOLUME WAS ABANDONED

The relevant source volume has been abandoned. Reasons for this vary and can be found in the "Error Messages" section above. In this circumstance, COPY goes to an abnormal end of job.

UPDATED BY MERGE

The requested file was not copied due to use of the keyword MERGE. Another copy of this file was found on a subsequent source, and therefore this file was "updated" by that file, and only that file was copied to the destinations.

FILE INCOMPLETE DUE TO PARITY ERROR

While copying a file from tape a parity error occurred. If the user opted to "CONTINUE" this file, the portion that had been copied prior to the error is left on the destination, but the file is incomplete.

I/O Error Handling and Messages

You will be notified by SYSTEM/COPY through the system ODT when unrecovered or internal errors occur. An internal error is one that is recognized only by SYSTEM/COPY. A parity error, on the other hand, is first handled by the operating system.

I/O errors will be retried in accordance with established operating system guidelines and, pending non-recovery, control will pass to SYSTEM/COPY for interpretation and possible action on your part.

The form of each of these error messages is illustrated in Figure 2-2 below:

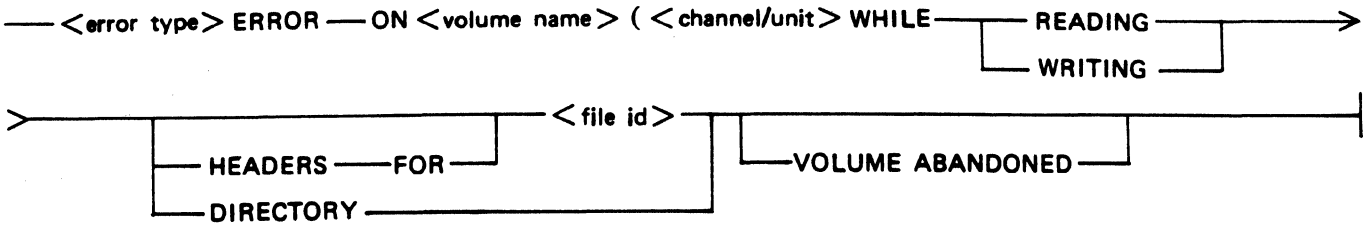


Figure 2-2. I/O Error Message Format

The <error type> defines the error. The <volume name> specifies the volume on which the error occurred; and when appropriate, the <channel/unit> is also given.

Below is a list and a brief explanation of each possible <error type>.

COUNT ERROR

Each block on a SYSTEM/COPY tape is numbered to ensure the proper read back sequence. This error indicates a violation of that sequence. The error may be caused by an undetected read or write, dropped or added tape blocks, or data corruption. In any of these circumstances, COPY goes to an abnormal end of job.

PREMATURE EOF

A premature end of file condition was encountered during read or write of this file.

PARITY ERROR

An unrecoverable parity error has occurred. When this happens, COPY goes to an abnormal end of job.

RELIABILITY ERROR

This error occurs when copying from a tape source that had an error when it was created that resulted in the file being abandoned. In this circumstance, COPY goes to an abnormal end of job.

Additional Error Possibilities

Both TAPE and ICTAPE volumes present additional error possibilities related to their headers, which precede the file and store information about the file. If an error occurs during read or write of this header information, you are notified by the HEADERS FOR phrase.

Similarly, each tape volume has a directory near the front of each reel. Errors encountered during a read or a write of this information are signaled by the DIRECTORY message. If an error occurs while the Directory is being read, there are no recovery options. Instead, the volume is abandoned at this point.

Recovery Options

In addition to the message displayed, you may be given recovery options. SYSTEM/COPY judges the severity and recoverability of the error to determine what possibilities to offer. These options follow the

error message and take one of the following three forms:

AX "CONTINUE" to ignore error
AX "QUIT" to abandon this file only
AX "ABORT" to abandon this volume

The response may be abbreviated to the first letter and will be checked for validity.

The CONTINUE option instructs SYSTEM/COPY to ignore an error and to continue with the operation. If tape destinations are involved, the block containing the error will be marked as unreliable. For this reason, a reliability error will be noted if the volume block is used as input. Any disk or diskpack destination files will be properly closed, but will not be complete. The user will have to determine the reliability of the file.

The QUIT option will stop transfer of this file and continue with the next file if one exists. Any destination tape files will be marked, and a subsequent usage will result in non transfer for this file. Any disk or diskpack destination files will be purged.

The ABORT option performs all the functions of the QUIT option, and also skips any remaining files requested from the current source volume.

Reliability Handling and Messages

Each block on a SYSTEM/COPY generated TAPE and ICTAPE contains reliability information. This information consists of a block count and a data integrity flag.

The block count begins at 1 and is incremented by 1 for each block of the file as it is written to the tape. When the tape is read, either during the compare phase or during the subsequent use of the tape, this sequence will be verified. If a discrepancy is found, a COUNT error will occur.

The data integrity flag indicates the validity of the information contained in this block. If an error occurred on the source file when this tape file was created, you will be given various options. Your response will be reflected in the value of the integrity flag written to the tape. This value is then interpreted when the tape block is read, resulting in the following message:

BLOCK NUMBER <number> ON TAPE <tape name>
CONTAINS UNRELIABLE DATA FOR SECTORS <sector range> OF FILE <file name>

The message tells you that an an error occurred when the tape file was created. It gives the block number and sector range where the error occurred. Another message may be displayed indicating what the final outcome of this error was. The form of each of these block error messages is illustrated in Figure 2-3 below:

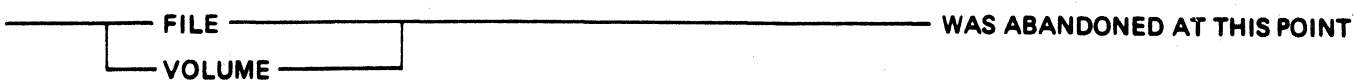


Figure 2-3. Block Error Message

This message describes your response to the error. If you respond with a CONTINUE, then the ABANDONED message will not appear.

The "FILE WAS ABANDONED" message indicates that you abandoned the file by using the QUIT response.

Finally, the "VOLUME WAS ABANDONED" message indicates that you abandoned this file and all files requested from that source volume by using the ABORT response.

ABANDONED results in all destination files being purged. Any tape destination files will be marked with the same reliability information.

Missing File Handling and Messages

SYSTEM/COPY may encounter a variety of missing file conditions during execution. Generally, these will be missing diskpacks, missing disk or diskpack files, and missing tape files, as well as duplicate disk and diskpack files. Each of these is described below.

Missing Diskpacks

When you are creating destination tapes, you need to obtain directory information before any file transfer can occur. SYSTEM/COPY assumes the disk directory to be accessible, but a diskpack may be off-line when its directory is needed. If this condition occurs, SYSTEM/COPY will respond with the following message:

<volume name> diskpack NOT ON LINE

You will then have the option to mount the required diskpack or ignore the request for this source volume. These two options are displayed as follows:

AX "CONTINUE" ONCE ON LINE, AX "ABANDON" TO ABANDON THIS VOLUME

Your response may be abbreviated to the first letter and will be checked for validity. The response will also be reflected on the summary if the summary option is set.

Missing Disk and Diskpack Files

Missing disk and diskpack file messages can be sent when the directory is being searched or after the directories have been searched as the files are about to be opened for transfer.

If a file is missing, or has been removed from the disk or diskpack after being located, SYSTEM/COPY will send the following message:

<file name> NOT FOUND ON <volume type> <volume name>

If the MANDATORY option has been set, SYSTEM/COPY displays the following message:

<count> FILE(S) NOT FOUND ON <volume type> <volume name> PLEASE AX TO ACKNOWLEDGE AND CONTINUE

SYSTEM/COPY halts until you acknowledge that some files have not been found. If you respond with an AX, SYSTEM/COPY will continue with the next file.

Missing Tape Files

When working with source tape volumes, the directory must be input and searched for all requested files. If you request more than one file for a destination of disk or diskpack only, their order may be altered to coincide with their order on the directory to avoid rewinding the tape. As with disk and diskpack files, SYSTEM/COPY will ensure that requested files are present and notify you of their absence.

Sometimes when a tape is created, a file may appear in the tape volume directory without actually being down tape. An example of this would be a parity error on the tape when it was created located after the directory but before the file was transferred. If the volume was abandoned at that point, the file would be in the tape's directory but not down tape. SYSTEM/COPY will notify you of this situation with the following message:

FILE NOT FOUND DOWNTAPE ON TAPE <tape-name>: <file-name>

Duplicate Disk and Diskpack Files

When you copy to disk or diskpack, and when a file with a matching ID is already present, the existing file will be replaced with the new file. If the existing file is in use when the new file is copied, the existing file is removed when it is no longer in use. If another person tries to access the existing file while it is still in use, the person will receive the newly copied file, not the existing, or original, file already on disk or diskpack.

Non-library Tape Handling and Messages

If you provide tape as input to SYSTEM/COPY, and if the tape is not a SYSTEM/COPY, LOADMP, or PACKUP format tape (that is, if it is a tape created by DMPALL, or is a "foreign" tape), then the following message will be displayed, and SYSTEM/COPY will go to an abnormal EOJ.

<volume name> (cc/uu) IS NOT SYSTEM/COPY, LOADMP OR PACKUP TAPE — ANALYSIS ABANDONED

LOADMP/PACKUP Tape Handling and Messages

If you use the "VOLUMEINDEX" option with a LOADMP or PACKUP tape, then the following message will be displayed, and SYSTEM/COPY will go to an ABNORMAL EOJ.

<volume name> (cc/uu) IS A LOADMP OR PACKUP TAPE
VOLUMEINDEX SPECIFICATION IS NOT ALLOWED — ANALYSIS ABANDONED

ICTAPE Format

The ICTAPE format follows the Unisys Corporate Standard format for interchangeable media. At this time however, neither B 1000 Series or the A Series implement this standard. It is a multi-file tape format. The general format of an ICTAPE is:

VOL1...file(s)

VOL1 represents the volume label written to the tape by the MCP. In this case, a file is of the general form:

HDR1...HDR2...TM...FID...data block(s)...TM...EOF1...EOF2

HDR1 and HDR2 are 80-byte records that are written by the MCP at file open time. The MCP receives the information for the HDR1 and HDR2 records from SYSTEM/COPY. These records do not contain the file header information that is kept in a disk or diskpack file header. The HDR2 record contains the maximum blocksize that occurs on the tape.

TM represents a tape mark.

FID, or file description block, contains the information found in a disk or diskpack file header. Both an object header (system dependent) and a symbolic header (system independent) are contained in the FID.

The FID and all data blocks are 9KB blocks unless the BUFFERSIZE option was used to override this default, plus 12 bytes of Block Information, which is described in more detail below. The last block of a file may not have a full 9KB of valid data, but a 9KB block is still written. Actual file EOF must be determined from the file header information in the FID.

EOF1 and EOF2 are 80-byte records that are written by the MCP at file close time, and they are essentially duplicates of the HDR1 and HDR2 records.

The first complete file on a tape is the DIRECTORY. The format of the DIRECTORY is:

HDR1...HDR2...directory block(s)...EOF1...EOF2

where the first directory block consists of:

FORMAT ENTRY...FILE ENTRIES

Any subsequent directory block(s) consist of only FILE ENTRIES. A FILE ENTRY of all zeros is the last entry.

Each block that is written to the tape contains a 12 byte Block Information Field. (This is referred to as the Record Information Field or RIF.) Because of this, each block is actually 9012 bytes, where the first 12 bytes are this field. These 12 bytes are identical to the first 12 bytes of the RIF for a TAPE format tape, which is defined later in this section.

Pos (UN)	Field Name	Len/Type	Content
1-20	transaction number	10UA	This field is a count on all data blocks in a file, starting with one for each file.
21-22	block type	1UA	A flag indicating the type of block: 1 = directory, 2 = FID, 3 = data.
23-24	reliability	1UA	Indicates the reliability of the data contained in this block.

Reading An ICTAPE

A program must specifically allocate an extra label area large enough to accommodate the USASI Standard Label that an ICTAPE has. That is, 28 bytes for a basic label area, and 80 bytes each for the HDR1, HDR2 records.

Opening the tape file with the multi-file ID set to the tape name, and the file-ID set to blanks results in the first file on the tape being opened. This will be TAPDIR for reel one, or the continuation of the file from the previous reel for any subsequent reels of a multi-reel set.

For any reel beyond reel 1, the first file-ID on the tape will be Fxxxxx, where xxxxx is a file sequence number relative to the first file on reel 1. This file is continued from the previous reel onto this reel. The next file will then be TAPDIR.

Each file on the tape is a separate file. Therefore, subsequent reads will read through the file, until EOF is sensed. Once the EOF has been sensed, the file can be closed. If the file-ID is reset to blanks, the next open will yield the next file. If you want a particular file, you can then request that file-ID, and the MCP will search the tape until the requested file is found.

You should perform file opens and closes with the no-rewind option if you want to read through the entire tape. A simple close will rewind the tape but leave the program attached. A close release will rewind the tape and return the unit to the system.

TAPE Format

The TAPE format is somewhat similar to the LOADMP/PACKUP tape format in that it is a single file tape. The general format of a TAPE is:

VOL1...HDR1...HDR2...TM...file(s)...TM...EOF1...EOF2

where a file is of the general form:

FID...data blocks

VOL1 represents the volume label written to the tape by the MCP.

HDR1 and HDR2 are 80-byte records that are written by the MCP at file open time. The MCP gets the information for the HDR1 and HDR2 records from SYSTEM/COPY. These records do not contain the file header information that is kept in a disk or diskpack file header. The HDR2 record contains the maximum blocksize that occurs on the tape.

TM represents a tape mark.

The file description block (FID) contains the information that is found in a disk or diskpack file header. Both an object header (system dependent) and a symbolic header (system independent) are contained in the FID.

The FID and all data blocks that are not a full block are written as short blocks to the tape. Though the exact size of a given data block is not stored, data block size can be determined from FIBRAD after a read. Actual file EOF must be determined from the file header information in the FID.

EOF1 and EOF2 are 80-byte records that are written by the MCP at file close time, and are essentially duplicates of the HDR1 and HDR2 records. The first file is a directory and has the general format:

directory blocks

where the first directory block consists of:

FORMAT ENTRY...FILE ENTRIES

Any subsequent directory blocks consist only of FILE ENTRIES. A FILE ENTRY of all zeros is the last entry.

Directory blocks are also written as short blocks if they are not a full 9KB.

The definitions of these fields are identical to the description in this document, with the following exception: the fifth byte in the HDR2 record is a "U," which designates variable length records, where length is determined by each I/O statement.

Each block written to the tape contains a 24 byte Record Information Field. This is referred to as the RIF. Because of this, each full block is actually 9024 bytes, where the first 24 bytes are this field.

The TAPE Record Information Field (RIF) is defined below.

Pos (UN)	Field Name	Len/Type	Content
1-20	transaction number	10UA	This field is a count on all data blocks in a file, starting with 1 for each file.
21-22	block type	1UA	A flag indicating the type of block: 1 = directory, 2 = FID, 3 = data.
23-24	reliability	1UA	Indicates the reliability of the data contained in this block.
25-36	checksum field	12UN	Used explicitly by DMCOPY (DMSII).
37	checksum flag	1UN	Used explicitly by DMCOPY (DMSII).
38	dmcopy tape type	1UN	Used explicitly by DMCOPY (DMSII).
39-48	filler	10UN	Reserved for future use.

Reading a TAPE

A TAPE is one continuous physical file. To logically differentiate one file from another, different block types have been defined. These block types are designated in a field in the RIF (as described above). Directory blocks will have a 1 in this field. FID blocks will have a 2 in this field. Data blocks will have a 3 in this field.

A program must specifically allocate extra label area large enough to accommodate the USASI Standard Label that a TAPE has. That is, 28 bytes for a Basic Label Area, and 80 bytes each for the HDR1, HDR2 records.

Opening the tape file with the multi-file ID set to the tape name, and the file-ID set to blanks will result in the first (and only) file on the tape being opened. This file-ID will be TAPDIR for reel one, or Fxxxxx, where xxxxx is a file sequence number, for any reel beyond reel 1. Fxxxxx indicates this is not reel 1, and that the first logical file is a continuation of the file from the previous reel. The next logical file will then be TAPDIR.

After file open, the first read will yield a type 1, or directory block (for reel one). Since there may be more than one directory block, the end of the directory can be determined by finally reading type 2 block (FID).

The FID will be followed by any number of type 3 data blocks. When another type 2 block is read, then the end of the previous file can be determined. This procedure is repeated for each file on the tape. The end of file for the last file will be determined by an actual EOF on the read.

Programmatically Differentiating TAPE, ICTAPE, LOADMP and PACKUP Tape Formats

You can differentiate ICTAPE, TAPE, LOADMP and PACKUP tape formats by first opening the tape.

The program must allocate extra label area large enough to accommodate the USASI standard label that a TAPE and an ICTAPE have. That is, 28 bytes for a Basic Label Area, and 80 bytes each for the HDR1, HDR2 records. LOADMP and PACKUP tapes have a standard label.

The tape file needs to be opened with the multi-file-ID set to the tape name, and the file-ID set to blanks. This results in the first file on the tape being opened.

At open time, the label area is filled.

If the HDR1, HDR2 records are not filled in, then the tape is a LOADMP or PACKUP tape. The eighth byte of the Basic Label Area will differentiate the tape format. For LOADMP it will be 5, and for PACKUP it will be 6.

If the HDR1, HDR2 records are filled in, then the tape is a TAPE or an ICTAPE. The fifth byte of the HDR2 record will differentiate the tape format. For a TAPE, this byte will be a "u", and for an ICTAPE it will be an "F".

Visually Differentiating TAPE, ICTAPE, LOADMP and PACKUP Tape Formats

For a TAPE, the first (and only physical) file on the tape is named TAPDIR. For an ICTAPE, the first file is TAPDIR. For any reel other than reel one of a multi-reel set of tapes, the first file on an ICTAPE will have a file-ID of Fxxxxx, where xxxxx is a file sequence number relative to the first file on reel 1. This file is continued from the previous reel onto this reel.

The first (and only) file on a LOADMP or PACKUP tape is named FILE.

An "OL MTP" when a tape is mounted and ready will give the multi-file-ID and the first file ID. For TAPE and ICTAPE format tape, the response looks like this.

<multi-file-ID>/TAPDIR for SYSTEM/COPY TAPE or ICTAPE

<multi-file-ID>/Fxxxxx for SYSTEM/COPY ICTAPES (reel > 1)

For LOADMP/PACKUP tapes, the response looks like this:

<multi-file-ID>/FILE for LOADMP/PACKUP tapes

To determine the exact format of a tape, TAPE, ICTAPE, LOADMP or PACKUP, you should use the SYSTEM/COPY DIRECTORY command to get a directory listing of the tape. The printer backup file generated will describe the exact tape format.

SECTION 3
DATA COMMUNICATIONS PROCESSOR (DCP)

A Data Communications Processor (DCP) enhances the data communications capabilities of the V Series systems. A DCP reduces the work load on the host system by performing data communication tasks that the host would otherwise have to handle. Some of these tasks include message routing to terminals, error recovery, code translation, line management, and network management.

System Requirements

A data communications system that uses a B 874 or a B 974 DCP requires an message control system (MCS), a network definition (NDL), and the DCP module of the MCP.

The MCS lets you manage various aspects of the data communications system, such as security, file control, error handling, audit and data recovery, message routing to applications programs, and resource allocation. An MCS (for example, GEMCOS) can be supplied by Unisys, or you may write your own in a high level language such as COBOL or BPL. Refer to the B 874/B 974 MCS Message Header Programming Reference Manual for more information.

The B 874 or B 974 Network Definition Language (NDL) source program includes a concise description of the network and of the procedures you should follow to control the network. The NDL source is normally written by you so that it meets your local network requirements. This NDL source is compiled with the B 874 NDL Compiler (CPNDL1) or the B 974 NDL Compiler (NDL974). The following files are a result of the compilation process: one NDL firmware file for each DCP, an MCP Network Information File (MCPNIF), and an MCS Network Information File (MCSNIF). These files are maintained on disk on the host.

The MCPNIF must be present on disk when you set the DCP option on the host system. The NDL programmer selects the name of the firmware file. The names of the NIF files are: MCPNpF and MCSNpF where the p represents the processor number of the host system processor that will use the files. If you want a B 874 and a B 974 DCP to run simultaneously on the same system, you must use the NIFMRG program to merge the NIF files.

You must load the firmware file into the DCP before it can operate. There are two commands you may use to load the firmware file into the B 874 or B 974. These are LH and SO DCP. Refer to the *V Series MCP/VS System Software Operations Guide, Volume 2: System Commands* for more information about the commands LH and SO. You specify DCP when you enter the command SO. Refer to the *V Series MCP/VS System Software Operations Guide, Volume 1: Installation* for more information about the USE DCP record.

Network Initialization

You should use the following procedure to initialize a data communications network to run a B 874 or a B 974 Data Communications Processor.

1. Perform a Cold Start on the system and be sure to declare the channel for each DCP. Each DCP must have its own UNIT card. Refer to the *V Series MCP/VS System Software Operations Guide, Volume 1: Installation* for more information about the DCP UNIT record.

2. Use the appropriate NDL compiler to generate a firmware file for each DCP and the MCPNIF and MCSNIF files. A B 874 and a B 974 DCP cannot run simultaneously on the same system until you use the NIFMRG program to merge the NIF files.

3. Set the DCP option with the command SO DCP. A command LH is done automatically with SO DCP.
4. Execute the MCS when the LDHOST program goes to EOJ or the B974LD program displays the following message:

   DATA COMM TO STATIONS HAS BEEN SUCCESSFULLY ACTIVATED

MCP Interface for B 874 or B 974

The following MCP records and commands initialize and control DCPs. For more information about UNIT and LIMIT records, refer to the *V Series MCP/VS System Software Operations Guide Volume 1: Installation.* For more information about commands and control instructions, refer to the *V Series MCP/VS System Software Operations Guide Volume 2: System Commands.*

DLP record

Use this record to declare a DLP for the DCP.

DCP UNIT Record

The DCP UNIT Record configures the system for a DCP.

LIMIT DCPQUE Record

The LIMIT DCPQUE Record sets the maximum WRITE functions (type 30) that may be outstanding in the DCP for each station.

LIMIT DCPBUF Record

The LIMIT DCPBUF Record sets the default number of input messages that an MCS result pool can store.

Command SO with DCP

The command SO with DCP sets the DCP option and loads the DCP module into the host processor's memory. The command executes the LDHOST program for each B 874 and the B974LD program for each B 974.

Command LH

The command LH loads firmware and initializes DCPs.

BUFFER Control Instruction

The BUFFER control instruction specifies how many input messages an MCS result pool can store. BUFFER overrides the value you specified in the LIMIT DCPBUF Record, for a specific MCS.

The following tables list B 874 and B 974 error conditions.

Table 3-1. B 874 Link Errors

INV R/D	**R/D = <result-descriptor>**
CONTROL TIMEOUT	R/D = F000
HTMALFUNCTION	R/D = F800
B 874 MALFUNCTION	R/D = F700
S-MEM PARITY	R/D = CC00
MTEB	R/D = C400
MTEX	R/D = C200
HTLINK PARITY	R/D = D100
B 874 LINK PARITY	R/D = D200
INV S-MEM ADRS	S-PTR = <S-mem-address>
INV FUNCTION TYPE	FUNC = <function-number>
INV PSN	PSN = <PSN-value>

Table 3-2. B 974 Link Errors

INV R/D	**R/D = <result descriptor>**
HUB NOT READY	R/D = E000
* BUS PARITY ERROR	R/D = D000
* BUS PARITY ERROR	R/D = D300
DATA TIMEOUT	R/D = C010
HUB PARITY ERROR	R/D = C020
ACCESS DENIED	R/D = C040
HUB TIMEOUT	R/D = C050
PARTNER ERROR	R/D = C060
READ ON WRITE ERROR	R/D = C070
MEMORY PAR ERROR	R/D = C080
WRITE SYSTEM GONE	R/D = C090
EARLY TERMINATION	R/D = C0A0
LENGTH ERROR	R/D = C0B0
CONTROL XFER TIMEOUT	R/D = C0C0
LPC ERROR	R/D = C0D0
AMR WRITE ERROR	R/D = C0E0
MEM BUS PARITY ERROR	R/D = C0F0
DCP DID NOT RESP TO A TEST	

* The HC-2 DLP lacks the "non-standard" strap

Table 3-3. Firmware Interface Errors Detected by MCP

INV R/H ON READ, FUN/OFN	**<R/H-type> <orig-function>**
INV R/H TYPE, FUNC	<R/H-type>
INV R/H ORIG FUN, O FUNC	<function-number>
INV R/H LSN, LSN	<LSN-value>
INV R/H S-PTR , S-PTR	<S-mem-address>
INV R/H MCS# MCS#	<MCS-number>
NO R/H ORIG FUNC, FUN, OFN	<R/H-type><orig-func>
INV R/H TEXT SIZE, SIZE	<integer>

SECTION 4
DLP UTILITIES (DLPXNO AND DLPXCO)

DLPXNO and DLPXCO are two programs that dump and clear data link processors (DLPs) that hang due to irrecoverable I/O operations.

When the MCP is running, you should use the program DLPXNO. If the MCP is not up and you are working from a V 300, you can use the floppy disk based program DLPXCO. If the MCP is not up and you are working from a V 500, you can use DLPXCO from the hard disk of the maintenance processor.

When the MCP detects a hung DLP, it displays the following message:

```
cc/u <device mnemonic> NO RSLT DESC
<program-id> INV I/O DESC <file-id> <address> X<segment>
<program-id> DS OR DP
```

Both programs read the firmware in the DLP and dump it. Both programs can display the dump information on the screen and print it.

Since DLPXNO uses the services of the MCP, it creates a printer backup file of the dump information. This file can be printed or copied to a tape for later use. However, since DLPXCO runs without the support of the MCP, it dumps directly to the printer.

The print file of the DLP dump contains the following information:

- All error messages following the I/O in error
- An I/O trace of console DLP operations
- Logging information from the DLP or connecting module

Both programs can display the dump information on the screen. The screen display includes the following information:

- The stack, including microcode address
- The microcode word
- The status count of the DLP

Both programs display error messages if they encounter problems. A listing of these error messages, along with a brief description of each, is included at the end of this section.

This program allows you to clear another DLP before resuming normal operations. Note that after these programs finish, you must reload the firmware.

Executing DLPXNO

V 300 — If you are working from a V 300 processor, you must use the following syntax to identify the test bus channel you will use to clear the hung DLP before you can execute DLPXNO. Press the transmit key after you have identified this channel.

UNIT <channel>/1 NST

Where "channel" represents the system console's DLP number. For example:

UNIT 0/1 NST

V 500 — If you are working from a V 500 processor, you must use the following syntax to identify the test bus channel you will use to clear the hung DLP before you can execute DLPXNO. Press the transmit (XMT) key after you have identified the channel.

UNIT 83/0 NST

The channel number 83 is the channel number of the V 500 test bus. 83 is the DLP channel number.

From either processor, execute DLPXNO by performing the following steps.

First, inhibit the hung DLP so that other programs cannot access it while you are working with it. Enter:

XC+<DLP channel number>

Next, execute DLPXNO. Enter:

EX DLPXNO (,"ccxx","yyy")

Where:

cc represents the channel number of the test bus access

xx represents the hung DLP channel number

yyy represents the DLP type

DLP types are:

CFE - Common Front End device: HT, MTP, CRD, PCH, or TPR. This is the default.

SSP - Shared System Processor

PSM - Path Selection Module

For example:

EX DLPXNO (,"1003","SSP")

This example clears an SSP on channel 03 using the test bus interface of the console DLP on channel 10. When the DLP has been cleared, use the following command to return it to the system.

XC-<DLP channel number>

You can now load the firmware that is required by the DLP.

Executing DLPXCO

V 300 — If you are working from a V 300, use the following procedure to execute the floppy-based version of DLPXCO at the maintenance processor.

Press the ON LINE switch on the processor cabinet so the word "CONSOLE" lights up. Insert the floppy disk named CV321A in the processor's floppy disk drive. If you have set the OCS channel number with option SO OCS on PANDMV, enter the following command:

LOAD SYS DLPXCO; RUN

If you have not set the OCS channel number on PANDMV, issue the following command:

LOAD SYS DLPXCO; MEM SYS 34 = nn ; RUN

Where:

nn represents the OCS channel number you will use.

You can issue the LOAD, MEM, and RUN commands separately.

Press the ON LINE switch on the processor cabinet so that the word "ON LINE" lights up.

The system prompts you to enter the date, the time, and the printer channel number with the following prompts:

ENTER DATE: MMDDYY

ENTER TIME: HHMM

ENTER 2 DIGIT PRINTER CHANNEL:

You should enter the date (month, day, and year), the time (hour and minute), and the channel number of the printer that will print the dump. If you are not going to print the dump, or if the printer is not available, enter FF.

The system then displays the following prompt:

ENTER TB CHAN, TEST CHAN AND TYPE AS <BBCCTTT>

Where:

BB represents the DLP number of the test bus.

CC represents the DLP number to be tested and cleared

TTT represents the DLP type.

Enter the test bus channel number, the channel number of the hung DLP, and the DLP type.

DLP types are:

CFE - Common Front End device: HT, MTP, CRD, PCH, or TPR. This is the default.

SSP - Shared System Processor

PSM - Path Selection Module

For example, you could enter:

- LOAD SYS DLPXCO; RUN

- When execution of DLPXCO is complete, reload the firmware required by the DLP.

V 500 — If you are working from a V 500, issue the following command to execute DLPXCO from the hard disk of the maintenance processor.

LOAD PROGRAM DLPXCO SYS Y;

Following this command, enter RUN SYS and press GO,; or enter RUN SYS and press function key F8 [EVENT]. The system prompts you to enter the date, the time, and the channel number of the printer that will print the dump as shown below.

ENTER DATE: MMDDYY

ENTER TIME: HHMM

ENTER 2 DIGIT PRINTER CHANNEL:

Enter the date (month, day, and year), and the time (hour and minute). If you do not want to print the dump, enter FF in the printer channel field. If you want to print the dump, enter the channel number of the printer that will print the dump.

The system will display the following prompt:

- ENTER TB CHAN, TEST CHAN AND TYPE AS <BBCCTTT>

Where:

- BB represents the test bus number

  CC represents the DLP number to be tested and cleared

  TTT represents the DLP type.

- Enter the test bus channel number, the channel number of the hung DLP, and the DLP type.

DLP types are:

CFE - Common Front End device: HT, MTP, CRD, PCH, or TPR. This is the default.

SSP - Shared System Processor

PSM - Path Selection Module

When DLPXCO execution is complete, reload the firmware required by the DLP, then halt/load the system.

Error Messages

DLPXNO and DLPXCO can display the following error messages. Recovery procedures are explained when appropriate.

CONSOLE DLP CANNOT BE CLEARED OR IN BASE WITH MODULE BEING CLEARED

A Universal Console DLP must be manually cleared. Press the BASE CLEAR button on the DLP base. This procedure is normally performed by a Unisys Customer Service Engineer (CSE).

FAILED CONNECTING TO BASE

The console DLP operation to connect the base was unsuccessful.

FAILED DISCONNECTING

The previous console DLP operation to disconnect from the maintenance card of the base being tested was unsuccessful.

FAILED FINAL CLEAR OF DRIVERS

The previous console DLP operation to clear the maintenance drivers was unsuccessful.

FAILED INITIAL CLEAR OF DRIVERS

A console DLP operation to clear the maintenance drivers was unsuccessful.

FAILED INITIAL READ

A console DLP operation to read and clear the buffer was unsuccessful.

FAILED LOCAL CLEAR

The previous console DLP operation to clear the DLP was unsuccessful.

FAILED OBTAINING DATA

The previous console DLP operation to collect the logging information in the console DLP buffers was unsuccessful.

FAILED READ OF DATA

The previous console DLP operations to collect the logging information in the console DLP buffers was unsuccessful.

FAILED SETTING OUT

A console DLP operation to specify the unit to be logged and cleared was unsuccessful.

INVALID CLASS OF DLP: <type of DLP>

The type of DLP declared is not correct. Execute again with the correct DLP type.

TEST CHANNEL <declared channel> NOT XCED

The declared channel was not inhibited with the MCP command XC. Verify that the correct channel was entered, inhibit it if necessary, and then execute the program again.

TEST CHANNEL <declared channel> NON NUMERIC

The second two characters identifying the hung DLP were not numeric. Execute again with the correct test channel.

UC CHANNEL <console DLP channel number> NON NUMERIC

The first two characters of the console DLP channel number were not numeric. Execute again with the correct channel number.

UC CHANNEL <declared channel> NOT UC DLP

A check of the declared channel did not return an ID indicating that it is a console DLP. Execute again with the correct console DLP channel.

UC TEST BUS <UC DLP channel/1> NOT DECLARED

The channel specified as the UC DLP did not have unit 1 declared as NST. Verify that the UC DLP channel is declared and that the unit <UC DLP channel/1> is declared as NST; then execute again.

These messages require that you re-execute the operation that failed. Contact a Unisys Field Service Representative if the problem persists.

SECTION 5
FILE CONVERSION UTILITY PROGRAM (DMPALL)

DMPALL is a very flexible file handling utility program that allows you to list, copy, and convert disk, diskpack, tape, and card files. These DMPALL functions are organized into three major categories:

- Listing Function—This function lets you print the contents of any disk, diskpack, magnetic tape, or punched card file in either an alphanumeric, digit, or combined format. You can specify that the printing begin or terminate at any particular record; you can select the record at which to start and stop by record number or by testing the contents of the record.
- Media Conversion Function—This function lets you copy a data file, changing file-id, record-length, blocking-factor, and parity, and, if you want, copying the file to another hardware type. You can specify that this operation begin or terminate at any particular record; you can select the record at which to start and stop by record number or by testing the contents of the record.
- Miscellaneous functions, including:
  - Printing the contents of a printer backup tape produced by a Unisys A Series or B 5000/B 6000/B 7000 Series systems.
  - Printing object code disk files in hexadecimal format.
  - Listing the names of files that are on a multi-file tape, including those produced by the discontinued LOADMP or PACKUP library utility programs.
  - Printing the contents of a file that is on a tape produced by the discontinued LOADMP or PACKUP library utility programs.

You can initiate DMPALL with the PERFORM or PFM command, or with the EXECUTE DMPALL command. If you use the the PERFORM command, you should enter the parameters described in this section directly. If you use the EXECUTE DMPALL command, you can give it parameters through AX system commands, or from a card file, or from an executing program.

USING DMPALL TO LIST FILES

DMPALL is a flexible tool for listing the contents of files. The following restrictions apply to the DMPALL LIST operation:

- Magnetic tape file (MTP) is assumed if you omit the hardware type.
- You may enter only one KEY statement for each execution of DMPALL.
- The options SKIP and STOP take precedence over WHEN and UNTIL. SKIP, STOP, WHEN and UNTIL take precedence over KEY.

When DMPALL lists ASCII files, it automatically translates them to EBCDIC, unless you use the NT option. When DMPALL goes to end-of-job, a count of the records input and the records output are displayed on the ODT. The DMPALL options you selected are printed at the end of the listing.

Syntax for Listing Files with DMPALL

The syntax for operating DMPALL is shown in Figure 5-1. The syntax elements are discussed after the figure. You can request more than one operation by using the keyword THEN, and following it with syntax for further operations starting with PERFORM or PFM.

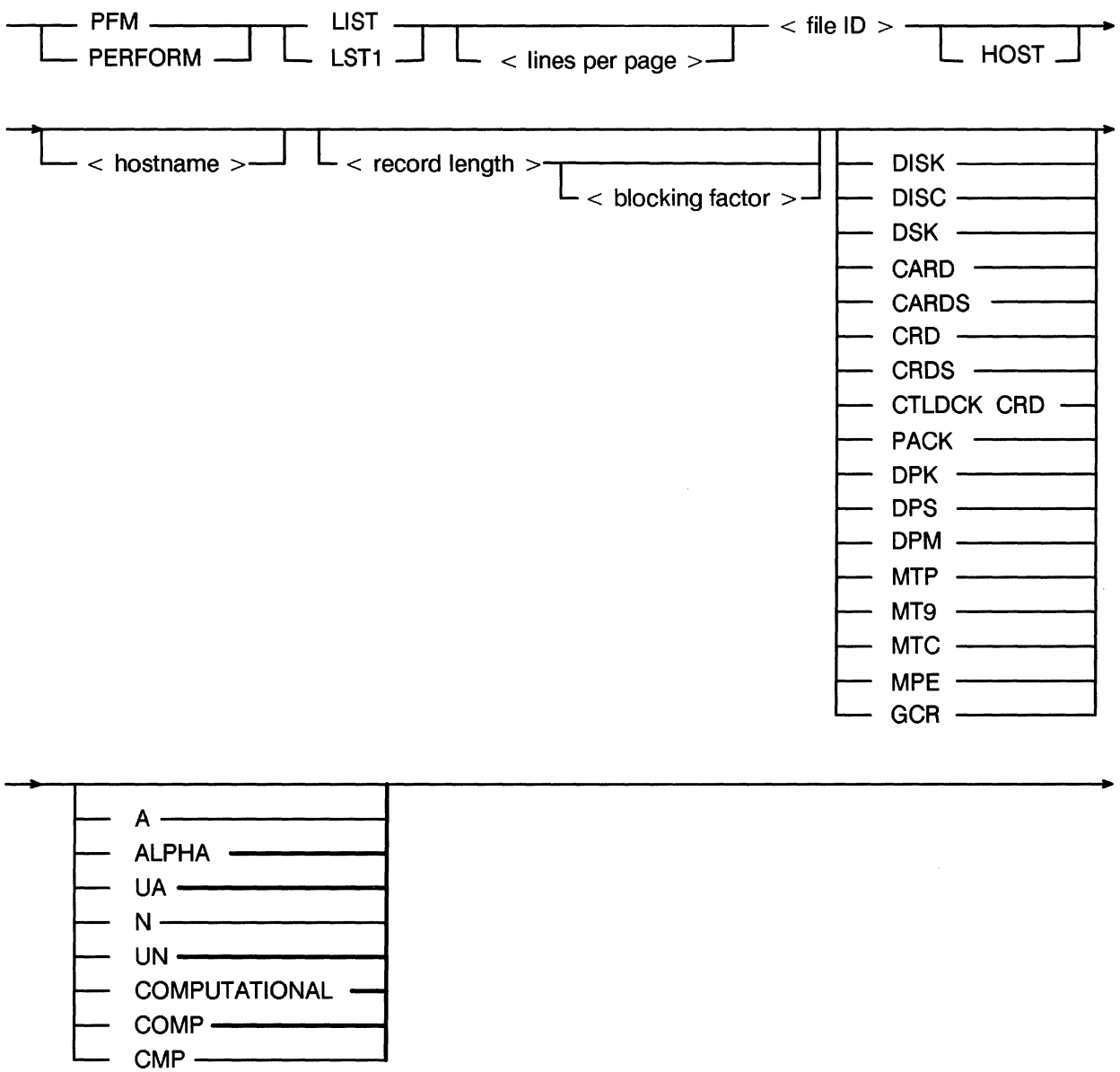


Figure 5-1. DMPALL File Listing Syntax

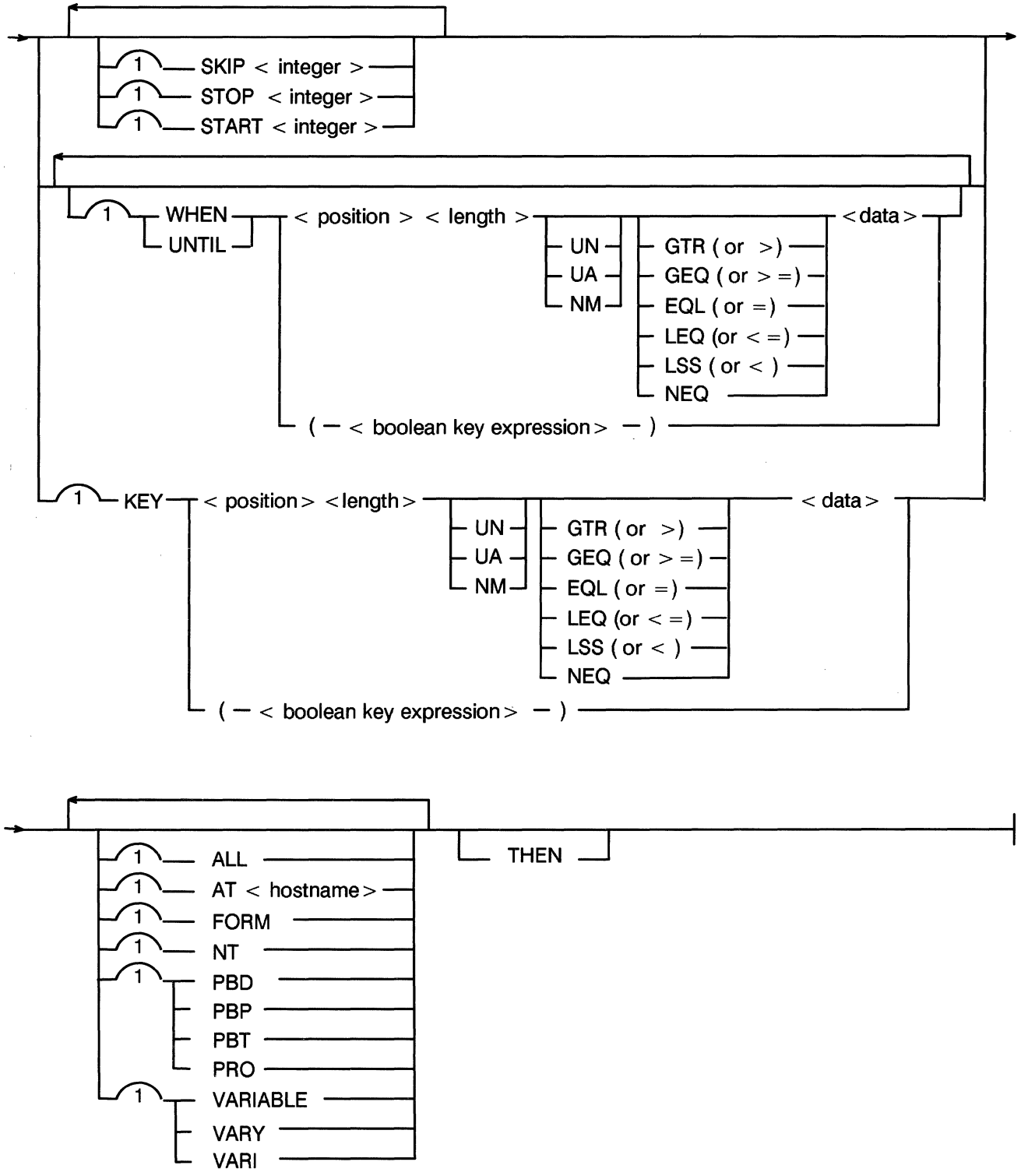


Figure 5-1 (Continued). DMPALL File Listing Syntax

Syntax elements for Listing Files with DMPALL

The elements that make up the DMPALL LIST syntax are defined below.

PERFORM
PFM

These keywords start the DMPALL program. They are synonyms.

LIST
LST1

These keywords inform DMPALL that you are listing a file, not converting one. If you enter LIST, the listing will be double-spaced. To get a single-spaced listing, enter LST1.

<lines per page>

You can use the <lines per page> entry to specify the number of lines for each page, excluding heading lines. The 12-punch on a carriage control tape, however, will override any lines per page entry that the 12-punch precedes. The default value for <lines per page> is 55 lines for each page.

<file ID>

The <file ID> is required. It is the name of the file you want listed. For multi-file tapes or diskpacks, you can format this parameter as <multi-file identifier/file-identifier>, where the <multi-file identifier > is the tape name or the diskpack family name. If the file name contains special characters, surround them with parentheses. The maximum length of the <file ID> is 8 characters — 6-character name plus 2 parenthesis characters.

<hostname>

This optional element indicates that the file to be listed resides on a remote host. The output listing is produced on the local host unless you enter the AT <hostname> element. If the hostname contains special characters, surround them with parentheses. The hostname parameter can consist of 1 to 17 alphanumeric characters, plus 2 parenthesis characters if needed.

<record length>

You can use the <record length> entry to format (unblock) tape files or to override the record length given in the disk or diskpack directory for disk or diskpack files. If you omit this entry, a <record length> of the memory size allocated to the DMPALL program minus 6000 digits is assumed for tape files. For disk or diskpack files, if you do not specify <record length> , the record length and blocking factor specified in the disk or diskpack directory are used. If you do specify the <record length>, be sure that it is an even number, denoting the length of a logical record in bytes. If you intend to use a < blocking factor>, you must always specify a <record length> first.

<blocking factor>

You use the blocking factor entry primarily to format (unblock) tape records or to override the blocking factor given in the disk or diskpack directory for disk or diskpack files. If you do not specify the value of <blocking-factor>, it is assumed to be 1, which indicates a single record for each block.

DISK
DISC
DSK

These are synonyms. They indicate that the file to be listed is on disk (100-byte-per-sector media).

CARD
CARDS
CRD
CRDS

These are synonyms. They indicate that the file to be listed is a punched card file. If there are any control cards in the card file, they will **not** be listed. A control card is a card with an invalid character (customarily a question mark, ?) in column one.

CTLDCK CRD

This indicates that the file to be listed is a punched card file. If there are any control cards in the card file, they **will** be listed. A control card is a card with an invalid character (customarily a question mark, ?) in column one.

PACK
DPK
DPS
DPM

These indicate that the file to be listed is a diskpack file. PACK is the default. DPK and DPM indicate that the output file can be spread over more than one spindle of the diskpack family. DPM indicates that the file must be confined to a single diskpack spindle.

MTP
MT9
MPE
GCR
MTC

These indicate that the file to be listed is a tape file. MTP is the default. MPE indicates phase-encoded recording mode. MT9 indicates NRZ recording mode. GCR indicates group character recognition recording mode. MTC indicates magnetic cartridge tape mode.

ALPHA
A
UA

These are synonyms, and list the file in alphanumeric (character) format. They are synonyms. Alphanumeric format translates your file into characters, so that you can read the text portions. You can put both this entry and the next entry (UN or N) to get a listing that includes both alpha and numeric formats; this is the default, and is good for files such as memory dumps that include both text passages and numeric passages.

UN
N
COMPUTATIONAL
COMP
CMP

These are synonyms, and list the file in numeric (digit) format. It prints the four-bit hexadecimal characters that make up the file, without providing a translation into alphanumeric format. This can be useful for a numeric data file. You can put both this entry and the preceding entry (ALPHA, A, or UA) to get a listing that includes both alpha and numeric formats; this is the default, and is good for files such as memory dumps that include both text passages and numeric passages.

START <integer>
SKIP <integer>

These are synonyms. They cause the listing to start with the record number you enter. You can enter any number from 0 to 99999999. If you omit this parameter or if the <integer> equals 1, printing begins with the first record in the file.

STOP <integer>

STOP causes the listing to stop after the record number you enter. You can enter any number from 0 to 99999999.

WHEN

WHEN causes the listing to begin with the first record that meets conditions you specify. The conditional statements are discussed under the KEY statement, below. You can set multiple conditions; all must be met before printing will begin.

UNTIL

UNTIL stops the listing of the file with the first record that meets conditions you specify. When the conditional statement becomes true, DMPALL stops printing the file, and then performs any request given by the THEN instruction. The conditional statements are discussed under the KEY function, below. You can set multiple conditions; all must be met before printing will stop.

KEY

You can use the KEY statement to print selected records. A record is selected for printing if a field within it meets the conditions you set. You select the field to be tested and set the conditions to be met by using the elements below.

The elements below are also used in the WHEN and UNTIL statements. In the KEY statement, you can set only one condition. In the WHEN and UNTIL statements, you can set several conditions, all of which may be met before the statement takes effect.

<position>

This gives the position, within the record, of the field that is to be tested. Give this parameter in digits or bytes, as determined by the UN, UA, or NM entry; default is digits. The test to be applied is given by the algebraic operators and the <data> statement. The first digit or byte of the record is digit or byte "1," **not** "0". The <position> is required.

<length>

This gives the length of the field you want tested. You enter this parameter in digits or bytes, as determined by the UN, UA, or NM element; default is digits. The maximum length is 40 digits or 20 bytes. The <length> is required.

UN
UA
NM

This designates the type of data you want tested, and also determines the meaning of the < position> and < length> elements. If you say neither UN, UA, or NM, this, UN is assumed.

UN means 4 bit unsigned numeric data; <position> and <length> are in digits.
UA means 8-bit unsigned alphanumeric data; < position> and < length> are in bytes.
NM means 8-bit signed numeric; < position> and < length> are in bytes.

GTR (or >)
GEQ (or >=)
EQL (or =)
LEQ (or <=)
LSS (or <)
NEQ

These algebraic operators determine the comparison that will be done between the designated field and the <data>. This is required. The operators, their meanings, and their alternate forms are shown below.

Greater than	GTR or >
Greater than or equal to	GEQ or >=
Equal	EQL or =
Less than or equal to	LEQ or <=
Less than	LSS or <
Not equal	NEQ

<data>

This is the value against which value in the specified field is compared. It is required.

<boolean key expression>

This is a set of comparisons enclosed within parentheses, separated by the boolean operators NOT, AND, or OR. Each comparison is composed of the elements: <position>, <length>, type of data, algebraic operator and <data>.

Using <boolean key expression>, you can direct DMPALL to perform more than one comparison. You can nest boolean key expressions within each other using extra pairs of parentheses. DMPALL uses the standard rules of boolean algebra to evaluate <boolean key expression>. You can enter a maximum of 20 comparisons.

Examples:

```
WHEN (0 4 UA EQL "TEST" AND 6 2 UN EQL 01)

KEY (NOT (73 5 UA EQL "ERROR" OR 73 5 UA EQL "DEBUG" ) )
```

ALL

ALL prints all the files on a multi-file tape, beginning with the first file you specify with the <file-identifier> entry.

AT < hostname >

AT <hostname> lets you specify the name of a remote host where you want the print file to be created. Keep in mind, though, that AT lets you specify only the destination hostname for the print file. That is, the input file you want listed is still assumed to reside on the local host unless you enter a hostname element after the < file ID> .

FORM

FORM notifies DMPALL that you want your listing printed on a special forms printer. You specify which printer with the FM system command, which is discussed in the *V Series MCP/VS System Software Operations Guide, Volume 2: System Commands*. You can abbreviate FORM as FM.

NT

NT indicates no translation. If this option is used, DMPALL will not convert ASCII to EBCDIC or EBCDIC to ASCII.

PBD

PBD directs the printer file to a printer backup disk.

PBP

PBP directs the printer file to a printer backup diskpack.

PBT

PBT directs the printer file to a printer backup tape.

PRO

PRO directs the printer file to a line printer. This is the default, unless overridden by a setting in the MCP configuration file.

VARIABLE
VARI
VARY

These are synonyms. You can use them to list a tape file that has variable record length, variable number of records per block, and variable block size. The first four bytes of each record must contain the length of that record in bytes. (This entry is not needed for a tape file that has variable record length but is unblocked or has one record per block.)

THEN

If you say THEN, you can input another string of DMPALL syntax. Follow the keyword THEN with syntax starting with PFM.

USING DMPALL TO CONVERT FILES

The syntax shown in Figures 5-3 and 5-4 shows how to direct DMPALL to convert a file from one storage medium to another, or to copy a file into another file with different characteristics on the same storage medium.

The following restrictions apply to this operation:

- The optional <file characteristics> for both input and output must be nested. That is, if the <blocking factor> is specified, the <record length> must be specified. If the <records per area> is specified, then the <record length> and <blocking factor> must be specified. If the <number of areas> is specified, then all <file characteristics> are required.
- The BIN routine type must not be used with pseudo readers, and is prohibited from going to punch backup. In addition, when reading binary cards, DMPALL can be terminated using either the QT System command, DS System command or a binary end card. The output file ID NONE is required when converting from binary to binary.
- The control parameter SORT can be specified only in a media conversion operation. The maximum allowable SORT statements are 10. The SORT statement will override any specified SKIP, STOP, WHEN, or UNTIL parameters in a media conversion operation.
- Random files with non-contiguous areas (for example, a file with 5 areas, in which only areas 1 and 5 are allocated), cannot be copied with the DMPALL program.

When DMPALL goes to end-of-job, the input record count, output record count, and blocking factor are displayed on the ODT.

Overview of Syntax for Converting Files with DMPALL

An overview of the syntax for using DMPALL to convert files is shown in Figure 5–2.

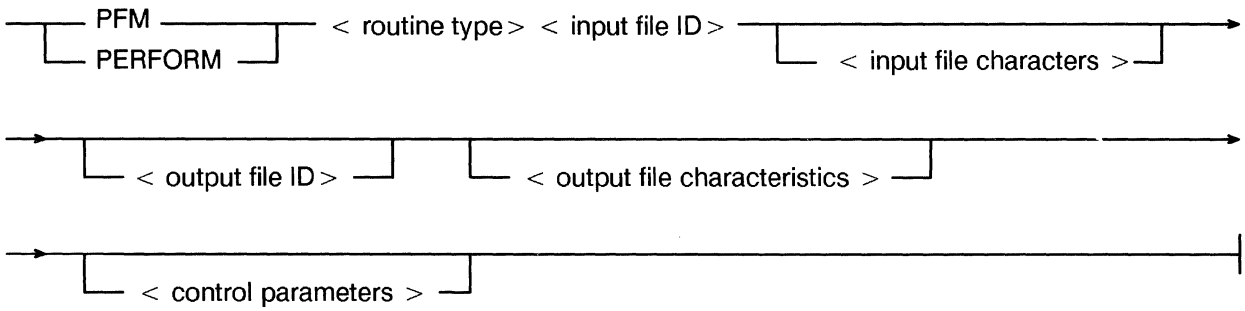


Figure 5-2. DMPALL File Conversion Syntax — Overview

Detailed Syntax for Converting Files with DMPALL

A detailed syntax for media conversion is shown in Figure 5–4. As shown in the diagram, you can repeat a looping process by selecting different elements for the same parameter. Limits for these processes are specified by the integer associated with the bridge, and cannot be exceeded. You can request more than one operation by using the THEN keyword, and then giving the syntax for another operation, starting with PERFORM or PFM.

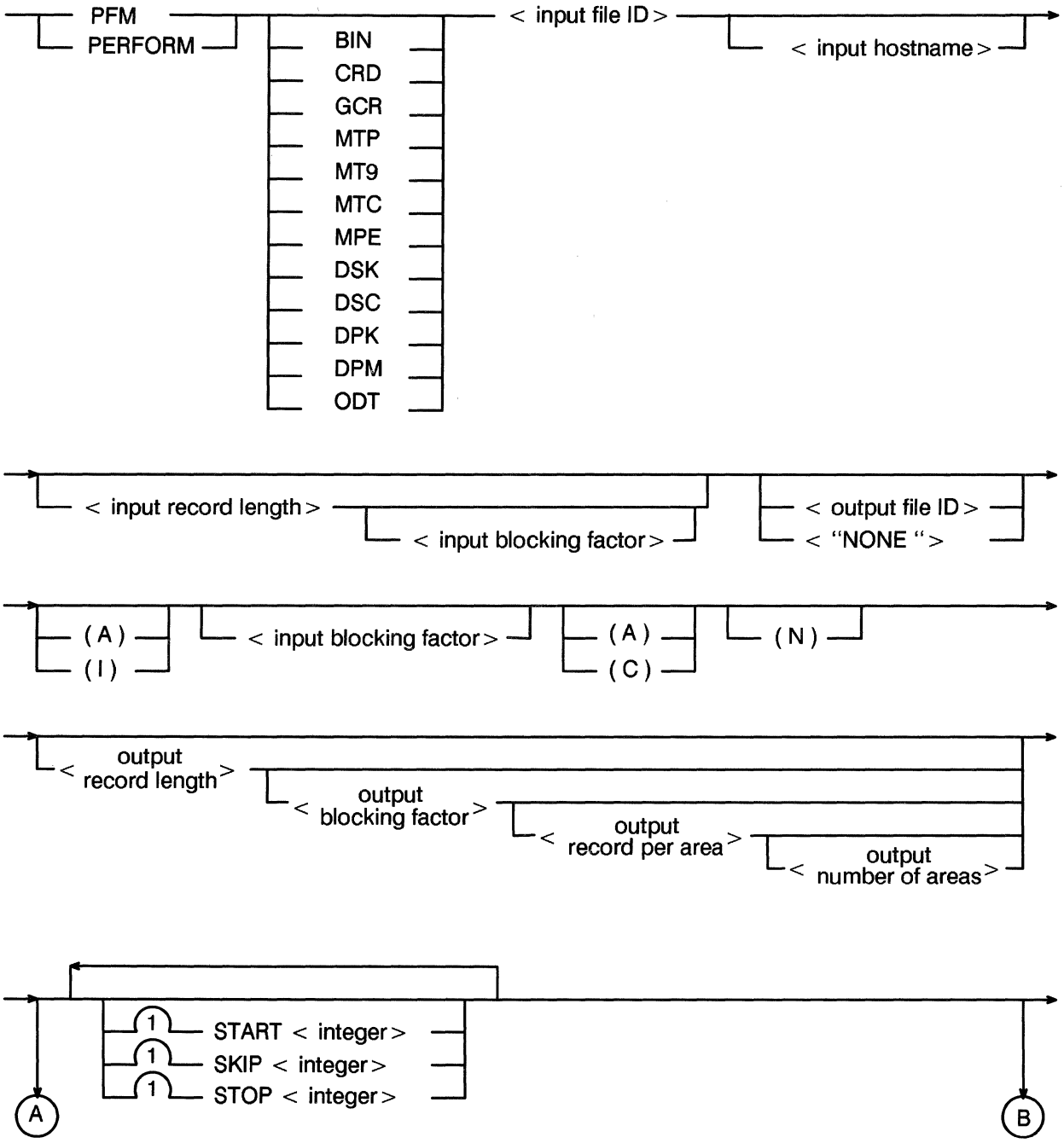


Figure 5-3. DMPALL File Conversion Syntax — Detailed

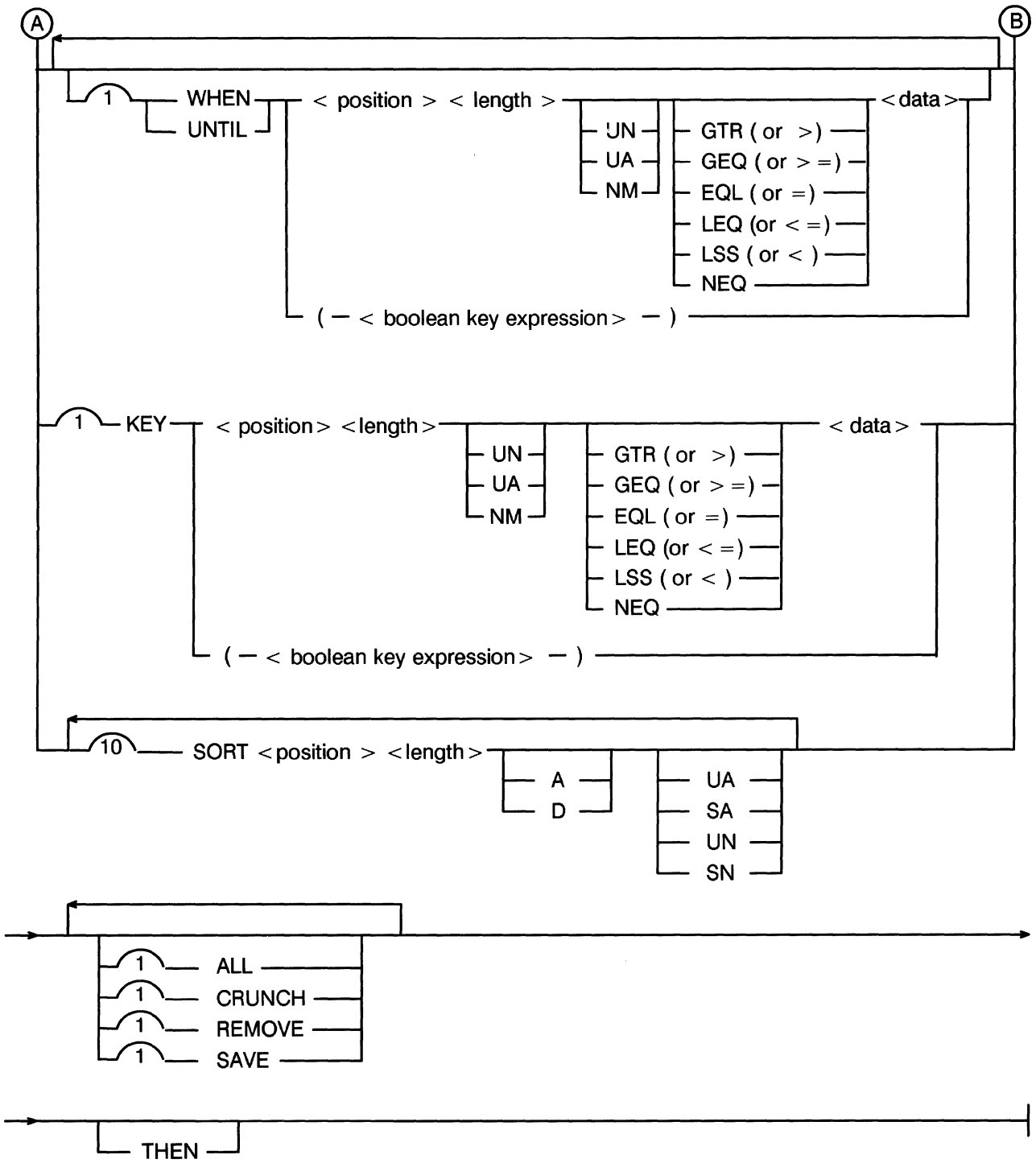


Figure 5-3 (Continued). DMPALL File Conversion Syntax — Detailed

DETAILED MEDIA CONVERSION PARAMETERS

Elements which appear in the overview diagram, Figure 5–2, are shown in **boldface** type.

PERFORM
PFM

This starts DMPALL.

<routine type>

The <routine type> is a required parameter tells DMPALL that you are converting a file, not listing one, and also tells what storage medium is the source and what storage medium is the destination. The <routine type> consists of a six-character concatenation of abbreviations for two hardware types. The first three characters specify the input device, where the file you want to convert resides. The last three characters to specify the output device, where you want the converted file to be placed. The various mnemonics you may use for this specification are listed here:

Binary cards (input or output)	BIN
Card reader (input) or card punch (output)	CRD
Card punch output	CPU
Magnetic tape (NRZ, phase-encoded, or GCR recording mode). This is the default.	MTP
Magnetic tape (NRZ recording mode)	MT9
Magnetic cartridge tape (GCR recording mode)	MTC
Magnetic tape (phase-encoded recording mode)	MPE
Magnetic tape (GCR recording mode)	GCR
Head-per-track-disk or LAK diskpack	DSK or DSC
Dispack (file can be spread over more than one diskpack spindle)	DPM or DPK
Diskpack (file is confined to a single diskpack spindle)	DPS
Operator Display Terminal	ODT

<input file ID>

This is the name of the file that is to be converted. It is required. If you do not want to use an input identifier, you should enter a dummy identifier, and you can then access the file with a UL system command. You must surround special characters with parentheses. The <input-file id> can be any identifier containing 1 to 6 characters, not counting any parentheses. For multi-file tapes or diskpacks, you may format this entry as <multi-file- id/file-id>.

<input file characteristics>

These optional entries describe the file that is to be converted.

<input hostname>

This optional entry indicates that the input file resides on a remote host. You can specify a hostname preceded by the key word HOST, HOSTNAME or HN, and an optional equal sign.

<input record length>

This optional entry is the record length of the input file, in bytes. The entry can be 1 to 6 digits long. The default record length for disk and diskpack files is the record length defined in the disk or diskpack header. For all other devices, the default is 80 bytes.

<input blocking factor>

This optional entry is the blocking factor of the input file. The value can be 1 to 3 digits. The default blocking factor for disk and diskpack files is the blocking factor defined in the disk or diskpack header. For all other devices, the default is unblocked files.

<output file ID>

This is the name that will be given to the converted file. Surround any special characters with parentheses. The file name can be from 1 to 6 characters long, not counting any parentheses. A multiple file entry is permitted for tape or diskpack files. If you omit this entry, the output file will be given the same name as the input file.

If you want to create an unlabeled file, enter "NONE", including the quotes, for the <output file ID >. This applies only to files other than disk or diskpack files.

<output file characteristics>

These optional entries describe the converted file you want DMPALL to produce.

(A)
(I)

This defines the file label you want on the output file. If you want to use an ANSI label, enter (A), including parentheses. If you want to use an installation defined label, enter (I), including parentheses. The use of an ANSI label or an installation-defined label only applies to output files written to devices other than disk or diskpack. The default is the Unisys V Series Standard Label.

If you want to create an unlabeled file, enter "NONE", including the quotes, for the <output file ID >. This applies only to files other than disk or diskpack files.

<output hostname>

This optional entry indicates that the output file will be placed on a remote host. You can specify a hostname preceded by the key word HOST, HOSTNAME or HN.

(A)
(C)

These determine how the file is assigned on disk or diskpack. Enter (A), including the parentheses, to have the file assigned by disk or diskpack areas. Enter (C), including the parentheses, to have the file assigned by diskpack cylinders; this choice is valid only if the destination is diskpack. The default is to assign the file by available space.

(N)

Enter an (N), including parentheses, to declare a nonstandard parity for the output file. If you

are using a BCL Punch file, you must specify (N).

<output record length>

This is the record length of the output file, in bytes. The value can be 1 to 6 digits. If you omit this parameter, the record length of the input file is used.

<output blocking factor>

This is the number of records per block for the output file. The value can be 1 to 3 digits. If you omit this entry, the blocking factor of the input file is used.

<output records per area>

This is the disk or diskpack area size, in records, of the output file. This entry is used only for disk and diskpack files. The value can be 1 to 8 digits. If you omit this entry, the output file will have the same number of records per area as the input file. If the input file is not a disk or diskpack file, and you do not give a value for < output records per area> , the default value of 500 records per area is used.

<output number of areas>

This is the number of disk or diskpack areas assigned to the output file. This entry is used for only disk and diskpack files. If you omit this entry, 20 areas are assumed unless the input is from disk or diskpack. If the input file is not a disk or diskpack file, and you omit this entry, the default value of 20 areas is used.

<control parameters>

These parameters let you choose which records to copy from the input file to the output file. The various <control parameters> are described below. Specific examples for the various <control parameters> appear later in this section under the heading "DMPALL Examples."

START <integer>
SKIP <integer>

These are synonyms. They cause conversion to start with the record number you enter, skipping over the previous records. You can enter any number from 0 to 99999999. If you omit this parameter or if the <integer> equals 1, conversion begins with the first record in the file.

STOP <integer>

STOP causes the conversion to stop after the record number you enter. You can enter any number from 0 to 99999999.

WHEN

WHEN causes the conversion to begin with the first record that meets conditions you specify. The conditional statements are discussed under the KEY statement, below. You can set multiple conditions; all must be met before conversion will begin.

UNTIL

UNTIL stops the conversion of the file with the first record that meets conditions you specify. When the conditional statement becomes true, DMPALL stops converting the file, and then performs any request given by the THEN instruction. The conditional statements are discussed under the KEY function, below. You can set multiple conditions; all must be met before conversion will stop.

KEY

You can use the KEY statement to copy selected records from the input file to the output file. A record is selected for conversion if a field within it meets the conditions you set. You select the field to be tested and set the conditions to be met by using the elements below.

The elements below are also used in the WHEN and UNTIL statements. In the KEY statement, you can set only one condition. In the WHEN and UNTIL statements, you can set several conditions, all of which may be met before the statement takes effect.

<position>

This gives the position, within the record, of the field that is to be tested. Give this parameter in digits or bytes, as determined by the UN, UA, or NM entry; default is digits. The test to be applied is given by the algebraic operators and the < data> statement. The first digit or byte of the record is digit or byte "1," **not** "0". The < position> is required.

<length>

This gives the length of the field you want tested. You enter this parameter in digits or bytes, as determined by the UN, UA, or NM element; default is digits. The maximum length is 40 digits or 20 bytes. The < length> is required.

UN
UA
NM

This designates the type of data you want tested, and also determines the meaning of the < position> and < length> elements. If you say neither UN, UA, or NM, UN is assumed.

UN means 4 bit unsigned numeric data; < position> and < length> are in digits.
UA means 8-bit unsigned alphanumeric data; <position> and <length> are in bytes.
NM means 8-bit signed numeric; < position> and < length> are in bytes.

GTR (or >)
GEQ (or >=)
EQL (or =)
LEQ (or <=)
LSS (or <)
NEQ

These algebraic operators determine the comparison that will be done between the designated field and the <data> . This is required. The operators, their meanings, and their alternate forms are shown below.

Greater than	GTR or >
Greater than or equal to	GEQ or >=
Equal	EQL or =
Less than or equal to	LEQ or <=
Less than	LSS or <
Not equal	NEQ

<data>

This is the value against which value in the specified field is compared. It is required.

<boolean key expression>

This is a set of comparisons enclosed within parentheses, separated by the boolean operators NOT, AND, or OR. Each comparison is composed of the elements: <position>, <length>, type of data, algebraic operator and <data>.

Using <boolean key expression>, you can direct DMPALL to perform more than one comparison. You can nest boolean key expressions within each other using extra pairs of parentheses. DMPALL uses the standard rules of boolean algebra to evaluate <boolean key expression>. You can enter a maximum of 20 comparisons.

Examples:

```
WHEN (0 4 UA EQL "TEST" AND 6 2 UN EQL 01)

KEY (NOT (73 5 UA EQL "ERROR" OR 73 5 UA EQL "DEBUG" ) )
```

SORT

You can use SORT to cause DMPALL to call the SORT. intrinsic with the parameters you specify to sort the input file records into the output file. You can enter multiple SORT statements to designate multiple sort keys. The first key you enter is the most significant. You can enter a maximum of 10 SORT statements. The final sequence of the sorted output file is based on the sort parameters shown below.

<position>

This required sort parameter indicates the position in the record of the field that is to be used as a sort key. You enter this parameter in digits if you specify UN or SN for the <type>, or in bytes if you specify UA or SA for the <type>.

<length>

This is a required SORT parameter that specifies the length of the field that is to be used as a sort key. You enter this parameter in digits if you specify UN or SN as the data type for the sort key in this SORT statement, or in bytes if you specify UA or SA as the data type for the sort key in this SORT statement.

ASCENDING (or A)
DESCENDING (or D)

This optional sort parameter determines the sequence of records in the sorted output file. The abbreviation for ASCENDING is A; the abbreviation for DESCENDING is D. The default is DESCENDING.

UA
SA
UN
SN

This optional sort parameter tells the data type of the sort key(s). The default is UN. The values are listed below:

8 bit unsigned alphanumeric	UA
8 bit signed alphanumeric	SA
4 bit unsigned numeric computational	UN
4 bit signed numeric computational	SN

ALL

You can use ALL during a tape to tape operation to copy all files on a multi-file tape. Copying begins with the file you identified in the <input file ID>.

CRUNCH

CRUNCH specifies that the output disk or diskpack file is to be closed with CRUNCH. This means that the file is condensed. All assigned but unused areas are discarded, and the <records-per-area> is reduced to the actual records used.

REMOVE

You can use REMOVE to close a disk and diskpack output files. This means that any existing file that has the same name as the output file on the destination medium will be automatically removed.

SAVE

You can use SAVE to cause a SAVE WITH LOCK operation on the output file if DMPALL is terminated abnormally.

THEN

If you say THEN, you can input another string of DMPALL syntax. Follow the keyword THEN with syntax starting with PFM.

MISCELLANEOUS DMPALL FUNCTIONS

The following commands are miscellaneous functions that you can perform using DMPALL. In this category are features incorporated for operator or programmer convenience.

Listing LOADMP and PACKUP Library Files

The PERFORM LIBLST function causes DMPALL to print the contents of a file from a tape produced by the LOADMP or PACKUP programs. (These programs are no longer supported by Unisys.) The format of the file you want to list must be the same as a printer backup disk file. The syntax for printing a file from these library tapes is as follows:

PFM LIBLST <tape ID> <file ID>

Example:

PFM LIBLST LIBT AAAA

Listing Object Code Files from Disk

The PERFORM LIST DSK function causes DMPALL to produce a hexadecimal listing of the object code disk file specified by <file ID>. This function also formats the segment dictionary, and indicates the location on disk of the beginning of each logical program segment. The <file ID> must refer to a disk-format object code file. The syntax for printing an object code file is as follows:

PFM LIST DSK <file ID> CODE

Example:

PFM LIST DSK AAAA CODE

Listing A Series Printer Backup Tapes

The PERFORM PB67 function causes DMPALL to print the contents of an A Series or B 5000/B 6000/B 7000 printer backup tape. The tape is must be labeled BACKUP. The correct channel/unit can be designated by an IL (Ignore Label) system command, discussed in the *V Series MCP/VS System Software Operations Guide, Volume 2: System Commands*. The syntax for printing a file from an A Series printer backup tape is as follows:

PFM PB67

Displaying Tape Directories

The SEARCH function lists the names of the files on a multi-file tape. This is primarily for use with multi-file tapes created by programs running on V Series systems. DMPALL locates a tape that has a tape name (multi-file identifier) that matches the <tape ID> you entered. DMPALL then scans the tape. DMPALL displays all the file names on the ODT, and lists information for each file. The syntax for listing the file names from a multi-file tape is as follows:

PFM LIST <tape ID > SEARCH

or

PFM LIST <tape ID > SEA

Example:

PFM LIST MYTAPE SEARCH

Listing LOADMP and PACKUP Tape Directories

The PERFORM PD function causes DMPALL to print a directory listing of a library tape produced by the LOADMP or PACKUP programs. (LOADMP and PACKUP are no longer supported by Unisys.) The syntax for printing a directory listing of these tapes is as follows:

PFM PD <tape ID>
or
PFM PD <channel number/unit number>

Examples:

PFM PD LIBT
PFM PD 6/2

ZIP Mechanism

ZIP is a programmatic interface for MCP system commands. The keyword ZIP followed by the name of a program causes the program to be started when DMPALL goes to end-of-job. Both of the following examples causes the program named PROGA to be scheduled for execution after DMPALL finishes.

Examples:

PFM LIST AAAA ZIP PROGA

The above example will start DMPALL, which will list the file called AAAA. When DMPALL finishes, the program called PROGA will start.

10AX DSKMTP AAAA ZIP PROGA

The above example can be used if DMPALL is executed with the EXECUTE command.

STARTING DMPALL WITH THE EXECUTE COMMAND

You can start DMPALL using the EXECUTE system command instead of the PERFORM command. You can enter the EXECUTE instruction from an ODT or a remote terminal, or from a card file or an executing program. When you start DMPALL this way, you give it parameters by using the AX (accept) system command discussed in the *V Series MCP/VS System Software Operations Guide, Volume 2: System Commands*. If you start DMPALL this way, you can utilize all of the parameters presented earlier in this section under "Listing Files with DMPALL" and "Converting Files with DMPALL." In addition , if you start DMPALL this way you can use a VALUE clause to access features that allow you to give commands to DMPALL from a card file or from an executing program. These VALUE clauses are discussed below.

How to Start DMPALL with the EXECUTE Command

The following procedures describe the necessary steps you should follow to start the DMPALL utility using the EXECUTE command from an ODT or a remote terminal. Starting DMPALL from a card file or an executing program is discussed under the VALUE clauses below.

1. At the ODT or a remote terminal, enter this command:

   EXECUTE DMPALL

2. DMPALL displays the following prompt:

   DMPALL = <mix no> ACCEPT

3. Enter an AX command of the form shown below, where <mix no> is the mix number of the DMPALL program as displayed in the ACCEPT message, and <options> are DMPALL parameters described earlier in this section under "Syntax for Listing Files with DMPALL" or "Detailed Syntax for Converting Files with DMPAL."

   <mix no> AX <options>

Example:

```
==> EX DMPALL

BOJ DMPALL=002 (MCP) 08:10

** DMPALL=002 ACCEPT

==> 2AX PFM LIST AAA STOP 100
```

For more information about the AX system command, see the *V Series MCP/VS System Software Operations Guide, Volume 2: System Commands.*

VALUE 0 = 100000

The EXECUTE DMPALL VALUE 0 = 100000 statement allows you to enter one PERFORM or PFM instruction to DMPALL through the card reader. You can enter the EXECUTE instruction through the ODT or in a card deck. When you perform this instruction, DMPALL searches for a card file with the file name of DMPALL and executes the instruction it contains. The first card in the file must say ?DATA DMPALL. The instructions can occur in any column on the cards. If there is more than one PERFORM or PFM instruction in the card deck, only the last one is used.

Below is an example of a card deck that starts DMPALL and copies a file from disk to tape (DSKMTP) with input (AAAA) and output (BBBB) having record lengths of 5000 bytes. The input disk file (AAAA) has 350 records per area with a blocking factor of 7. The output file (BBBB) is unblocked.

```
?EXECUTE DMPALL VALUE 0 = 100000
?DATA DMPALL
PFM DSKMTP AAAA 5000 7 350 BBBB 5000 1
?END
```

VALUE 0 = 000001

The EXECUTE DMPALL VALUE 0 = 000001 statement allows you to enter more than one PERFORM of PFM instruction to DMPALL at one time through a card reader. You can enter the EXECUTE statement through the ODT or a card deck in accordance with the following example. When you perform this function, DMPALL searches for a card file with the ID of DMPALL and then executes the instructions in that file. The first card in the file must say ?DATA DMPALL. The instructions can occur in any column position on the cards. If you use parentheses or quotation marks, the left and right characters must be on the same card.

The following example starts DMPALL and creates a multi-file tape (AFILE) with the files AAAA, BBBB, and CCCC being copied on the tape (AFILE). The output (tape) files will have the same record lengths and blocking factors as the input (disk) files.

```
?EXECUTE DMPALL VALUE 0 = 000001
?DATA DMPALL
PFM DSKMTP AAAA AFILE/AAAA
PFM DSKMTP BBBB AFILE/BBBB
PFM DSKMTP CCCC AFILE/CCCC
?END
```

VALUE 0 = 001000

The EXECUTE DMPALL VALUE 0 = 001000 statement allows DMPALL to receive data from another program through a FILL statement in languages that provide this capability. The FILL statement invokes the Core-to-Core interprogram communication function to obtain DMPALL parameters. DMPALL will do a Global FILL from any program. The program sending the DMPALL parameters must send them to the program named "DMPALL". It is the user's responsibility to provide the program to send the DMPALL parameters. The DMPALL parameters can start in any column, but can be no longer than 60 characters.

In the following example of a piece of a COBOL program, the ZIP statement will start DMPALL and inform DMPALL that the specifications will come from Core-to-Core. The FILL statement will then supply the specifications.

```
WORKING-STORAGE SECTION.
77 DMPALL-EXECUTE PIC X(72)
VALUE "EX DMPALL VA 0 = 1000.".
77 SPECS PIC X(60)
VALUE "PFM LIST MSTRFL DSK STOP 100.".
.
.
PROCEDURE DIVISION.
START-DMPALL.
ZIP DMPALL-EXECUTE.
FILL SPECS INTO "DMPALL".
```

WHEN DMPALL ENCOUNTERS A PARITY ERROR

When DMPALL encounters a parity error on an input tape, the system displays a warning message indicating the error, and then requests that the operator enter an AX or DS system command. Below is an example of the messages produced at the ODT by a parity error.

```
==> RP 6/4

06/4 MPE PURGED

06/1 MTP PAR ERR

DMPALL=14 DROPPED BLOCKING # 16

DMPALL=14 AX or DS
```

You can perform one of three operations when encountering a parity error.

1. You can terminate DMPALL by entering:

   <mix no> DS

   where <mix no>is the mix number of DMPALL, as displayed in the request for AX or DS.

   Example:

   14 DS

2. You can continue processing and ignore this parity error and all subsequent parity errors in this execution of DMPALL by entering:

   <mix no> AX IGNORE

   IGNORE can be abbreviated as I, IG, IGN, IGNO, or IGNOR.

   Example:

   14 AX IGNORE

   14 AX IGN

3. You can continue processing and ignore this specific parity error only by entering:

   <mix no> AX XXXXX

   If you use any word (customarily, XXXXX) except IGNORE or its abbreviations (I, IG, IGN, IGNO, or IGNOR), DMPALL will ignore only the first parity error. Any subsequent parity errors will cause DMPALL to stop. When DMPALL stops and issues a warning message, an AX or DS command is displayed. At this point, you can enter any response, including IGNORE, and DMPALL will continue processing.

   Example:

   14 AX XXXXX

DMPALL EXAMPLES

PFM LIST TAP1 ALL
PFM LIST TAP1/CCCC ALL

The first command in this example initiates the printing of all files on TAP1. The second command initiates the printing of all files on TAP1, beginning with file CCCC.

PFM LIST AAAA AT MCPSystemB

This command executes the printing of the AAAA file on another system (MCPSystemB).

PFM LIST AAAA FORM

This sample FORM command notifies DMPALL that special forms be printed on a specific printer through the FM system command.

PFM LIST AAAA HOST

This sample MCPSystemB HOST command indicates that the file AAAA resides on a remote host.

PFM LIST AAAA KEY 10 2 UA EQL AB

This sample command indicates that only those records in file AAAA that have AB in bytes 10-11 will be printed. The contents of the comparison is based in the KEY entry specified by <position> 10, for a field <length> of 2, and the specified <data> AB. The data <type> tested is defined as UA. The record is printed if the EQL condition <operator> is matched.

PFM LIST AAAA

This sample command indicates that records from the AAAA file are to be printed double-spaced.

PFM LST1 AAAA

This sample command indicates that records from the AAAA file are to be printed single-spaced.

PFM LIST AAAA NT

This sample command prints the contents of a magnetic tape with no translation (NT).

PFM LIST AAAA PBD

This sample command prints the file AAAA to a printer backup disk (PBD).

PFM LIST AAAA PBP

This sample command prints the file AAAA to a printer backup diskpack (PBP).

PFM LIST AAAA PBT

This sample command prints the file AAAA is to a printer backup tape (PBT).

PFM LIST AAAA PRO

This sample command prints the file AAAA to a line printer (PRO).

PFM LIST AAAA SKIP 100

This sample SKIP command causes printing to begin with the record number denoted by <integer> 100.

PFM LIST AAAA START 100

This sample START command causes printing to begin with the record number denoted by <integer> 100. If you specify <integer> 0, printing begins with the first record in the file.

PFM LIST AAAA STOP 30

This sample STOP command causes the first 30 records (records 1–30) of the AAAA file to be printed.

PFM LIST AAAA DSK STOP 30
THENPFM DSKMT9 AAAA MASTER 100 5

This sample command lists the first 30 records of the disk file AAAA. Then the file AAAA is written to a magnetic tape with a <file-id> of MASTER. According to this sample command, the output tape MASTER has a record length of 100 bytes, and a blocking factor of 5.

PFM LST1 AAAA VARIABLE A

This sample VARIABLE command produces a single-spaced listing (LST1), in alpha format (A), of the data on a variable record length tape labeled AAAA.

PFM LIST AAAA DSK WHEN 10
2 UA EQL AB

This sample command indicates that printing begins with the first record in the AAAA file that matches the WHEN parameter. The contents of comparison are based in the WHEN entry specified by <position> 10, for a field <length> of 2, and the specified <data> AB. The data <type> tested is UA. The record is printed if the EQL condition <operator> is matched.

PFM LIST AAAA DSK UNTIL 10
2 UA EQL AB THENPFM DSKMT9
AAAA BBBB 100 5

This sample command indicates that printing begins immediately, but stops when the first record in the AAAA file matches the UNTIL parameter. The contents of comparison are based in the UNTIL entry specified by <position> 10, for a field <length> of 2, and the specified <data> AB. The data <type> tested is UA. DMPALL stops the print process when the EQL condition <operator>is met. In this sample command, DMPALL will then perform the next function.

PFM MTPMTP AAAA ALL

This sample command initiates the copying of all files on AAAA.

PFM DSKDSK AAAA BBBB CRUNCH

This sample command closes and condenses the file BBBB by indicating CRUNCH.

PFM DSKDSK AAAA KEY 10 2
UA EQL AB

This sample command indicates that only those records in file AAAA that have AB in bytes 10-11 are selected. The contents of comparison are based in the KEY entry specified by <position> 10, for a field <length> of 2, and the specified <data> field AB. The data <type> tested is defined as UA. The record is copied if the EQL condition <operator>is matched.

```
PFM DSKDSK AAAA REMOVE
```

This sample command copies the file AAAA without changing the filename, and the old copy is automatically removed.

```
PFM DSKDSK AAAA SAVE
```

In this sample command if the file AAAA terminates abnormally, the entry SAVE will close the file with lock.

```
PFM DSKDSK AAAA SKIP 100
```

In this sample command SKIP causes the selection of records to begin at the 100 record on the AAAA file.

```
PFM DSKDSK AAAA BBBB SORT 1
7 D UA
```

In this sample command a disk file named AAAA will be sorted in descending sequence. The type of data in the SORT key is unsigned alphanumeric. The position of the SORT key is in the first byte of the record. The length of the SORT key is seven bytes. The sorted output disk file is named BBBB.

```
PFM DSKDSK AAAA START 100
```

In this sample command START causes the selection of records on the AAAA file to begin at record 100.

```
PFM DSKDSK AAAA STOP 500
```

In this sample command STOP causes the first 500 records of the AAAA file to be copied.

```
PFM DSKDSK AAAA STOP 100
THEN PFM DSKMTP AAAA MASTER
100 5
```

In this sample command the first 100 records of the disk file AAAA are copied to disk using the same file name. Then the AAAA file is copied to a magnetic tape with an <output-file-id>named MASTER. The output tape has a record length of 100 bytes, and a blocking factor of 5.

```
PFM DSKDSK AAAA WHEN 10
2 UA EQL AB
```

In this sample command copying begins with the first record in the AAAA file that matches the WHEN parameter. The contents of comparison are based in the WHEN entry specified by <position> 10, for a field <length> of 2, and the specified <data> AB. The data <type> tested i: UA. Copying begins when the EQL condition <operator> is matched.

```
PFM DSKDSK AAAA UNTIL 10
2 UA EQL AB THEN PFM
DSKMTP AAAA BBBB 100 5
```

In this sample command copying begins immediately, but stops when the first record in AAAA file matches the UNTIL parameter. The contents of comparison are based in the UNTIL entry specified by <position> 10, for a field <length> of 2, and the specified <data> AB. The data <type> tested is UA. DMPALL stops the copy when the EQL condition is met. DMPALL will then perform any <new-request>. In this example, the THEN function is executed.

SECTION 6
MEMORY DUMP UTILITY (DMPMEM)

In the event of a system failure, you can dump the MCP to tape. If you are working from a V 300 processor, you will use DMPMEM, the floppy disk based memory dump utility, to dump the MCP to tape. If you are working from a V 500 processor, you load the DMPMEM utility from the hard disk of the maintenance processor. DMPMEM is useful for abnormal failures. Typically, dumps for normal failures are performed by the Fault Handler.

DMPMEM is a control state program, and it is executed from the maintenance processor. Once DMPMEM has dumped the MCP to tape, the MCP dump can be analyzed by the DMPANL utility to help determine what may have caused the failure.

When the program writes to the tape, it first writes a header block, followed by a data block. The dump starts at the memory address 0 and continues to the end of MCP memory. The dump analysis utility DMPANL uses this header block for error detection. This header block describes the data block, which is an exact duplication of data contained in a segment of memory.

Executing DMPMEM

V 300 —If you are working from a V 300 processor, follow these steps to execute DMPMEM at the Universal Console.

If the system is not stopped, take the processor off line and put it in console mode. You can do this by pressing the ON LINE/CONSOLE switch on the processor cabinet so that the word CONSOLE on the switch lights up.

Insert the floppy disk named CV321A into one of the floppy disk drives in the processor cabinet and enter the following command:

```
LOAD SYS DMPMEM; RUN
```

Once DMPMEM executes, the system displays the following prompts, to which you can enter more information.

"Please enter tape drive channel & unit (cc/uu):"
"Please mount tape on cc/uu"
"Tape <tape number> mounted."
"O.K. to continue? Y/N"
"Memory dump to tape completed"

If DMPMEM cannot write to the tape, it will retry and display "Tape write retry."

Use another tape if this message appears frequently. If an unrecoverable error occurs, DMPMEM terminates instead of producing an unreliable tape.

After the program performs the dump, it terminates on a halt branch instruction (OP=29).

Refer to Section 22, DMPANL, for more information on analyzing MCP dumps.

V 500 —If you are working from a V 500 processor, enter the following command to execute DMPMEM at the maintenance processor.

```
LOAD DMPMEM;
```

Press the GO key, then press function key F8 [EVENT].

Once DMPMEM executes, the system displays the following prompts, to which you can enter more information:

"Please enter tape drive channel & unit (cc/uu):"
"Please mount tape on cc/uu"
"Tape <tape number> mounted."
"O.K. to continue? Y/N"
"Memory dump to tape completed"

If DMPMEM cannot write to the tape, it will try to write to the tape again and display the message: "Tape write retry."

Use another tape if this message appears frequently. If an error occurs from which you cannot recover, DMPMEM will terminate instead of producing an unreliable tape.

This utility terminates on a halt branch instruction (OP=29) after performing the dump.

Refer to Section 22 for more information on analyzing MCP dumps.

SECTION 7
DISKPACK UTILITY PROGRAM (DISPKV)

DISPKV is a multi-function utility that performs various disk and diskpack initialization functions. The specific functions performed during each execution are determined by the command and other supporting data input as a string through the ODT or on cards.

The major functions that can be performed by DISPKV include:

- Initializing virgin media as diskpack (180-byte sectors)
- Initializing virgin media as disk (100-byte sectors)
- Re-initializing previously initialized media (100- or 180-byte sectors) as diskpack (180-byte sectors)
- Re-initializing previously initialized media (100- or 180-byte sectors) as disk (100-byte sectors)
- Re-initializing a selected cylinder or a range of cylinders of a previously initialized disk or diskpack
- Relocating a selected sector on a disk or diskpack
- Verifying all sectors on a disk or diskpack
- Labeling a diskpack
- Changing selected fields in a diskpack label
- Purging a diskpack
- Listing the addresses of all relocated sectors on a disk or diskpack

EXECUTION SYNTAX

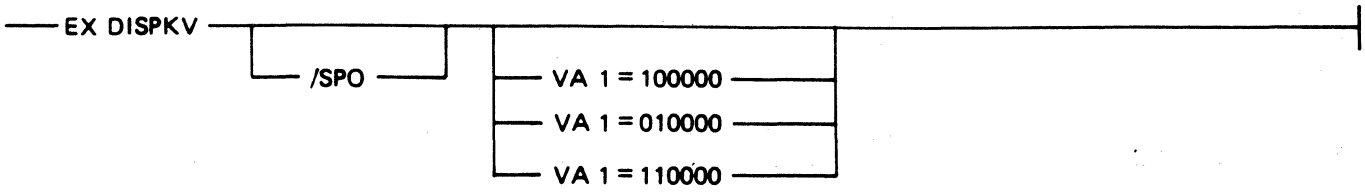


Figure 7-1. DISPKV Syntax

EX DISPKV

Initiates the DISPKV utility. Once DISPKV is initiated, parameters must be provided to indicate what you want it to do. The default input medium for these parameters is a card file called INPUTF.

(,"SPO")

Identifies that the parameters for DISPKV will be provided through the ODT using the AX system command.

If an error is encountered during ODT input, a detailed set of instructions appears explaining how to correct the input error.

VA 1=100000

This value statement sets switch 1 and must be used when performing the following functions:

- request a full IVR on a 235, 207, or 659 diskpack
- convert a 235 or 207 diskpack from 100-byte to 180-byte format
- convert a 235 or 207 diskpack from 180-byte to 100-byte format
- convert a 207 or 659 diskpack from sequential to interlaced mode
- convert a 207 or 659 diskpack from interlaced to sequential mode

Using this value statement also forces a short (one pass) Initialization, Verification, and Relocation (IVR) on all diskpack types except 207, 235, 659, 680, 682, MD4, and MD8.

VA 1=010000

This value statement sets switch 2 and is used to eliminate retries during verification. When switch 2 is set, any errors found during verification are immediately relocated and no retries are performed.

VA 1=110000

This value statement sets both switches 1 and 2 as described above.

The syntax for inputing DISPKV parameters through the ODT is shown in Figure 7-2.

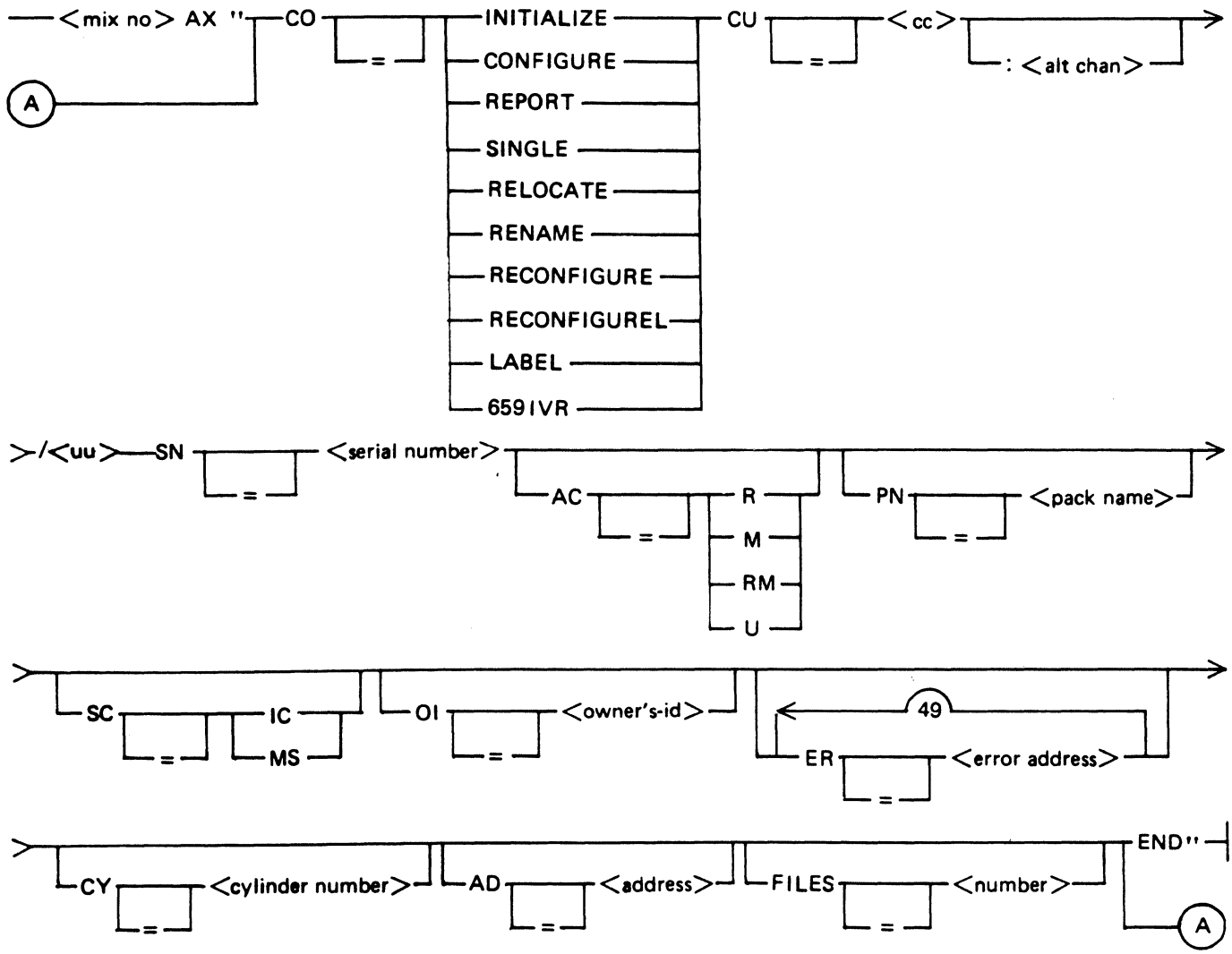


Figure 7-2. DISPKV General Syntax

<mix no> AX "

After DISPKV is executed using the (,SPO) option, parameters are input through the ODT using the AX (Accept) system command. The entire parameter string must be enclosed in quotation marks. All data input within the quotation marks is free form.

Each parameter is preceded by a two-letter keyword that identifies the parameter. An equal sign is optional between the keyword and the parameter. If the keyword is not followed by an equal sign, it must be followed by at least one blank.

Table 7-1 shows the DISPKV keywords, their meanings, and their default values (if any). If the default value is desired, the keyword associated with that value should not appear in the input parameter string.

Table 7-1. DISPKV Parameter Keywords and Default Values

Key Word	Meaning	Default
CO	Command	None
CU	Channel/unit	None
SN	Serial number	None
AC	System access code	U (Unrestricted)
PN	Pack name	SYS180
SC	System interchange code	MS (Medium Systems)
OI	Owner's-id	All blanks
ER	Error address	None
CY	Cylinder number	None
AD	Address	None
FILES	Files	110

CO

The CO keyword identifies the command parameter. All subsequent keywords and parameters, until the the next CO, are related to the command parameter and can be input in any order. Each command will be described in its own railroad diagram. Table 7-2 identifies the functions performed by each of the DISPKV commands. Examples:

CO=INITIALIZE ...

CO RECONFIGURE ...

SN <serial number>

SN identifies the serial number parameter. The serial number is a one- to six-digit numeric combination. It cannot be all zeros. Examples:

... SN=12345 ...

... SN 123456 ...

CU <cc>:<alt chan>/<uu>

CU identifies the channel/alternate channel/unit number parameter. These parameters must be numeric. Channel and unit are required while the alternate channel portion of the parameter is optional. Only one alternate channel can be declared in the CU parameter. If specified, the alternate channel declaration must be preceded by a colon.

The alternate channel declaration allows DISPKV to use an alternate I/O path to a diskpack subsystem. To declare an alternate channel to a diskpack, see the DISK, DLP and EXCHANGE records in the *V Series MCP/VS System Software Operations Guide, Volume 1: Installation.* After you execute DISPKV, you can then use this CU parameter to indicate that the diskpack can be accessed via the alternate channel. Since many DISPKV functions (especially INITIALIZE and CONFIGURE) do specialized I/O operations that can be quite lengthy, using the alternate channel would reduce competition for the primary channel.

The channel and alternate channel must be valid dispack DLPs, and the unit must be a unit number 0- 15. If an alternate channel is not specified, the primary channel is used. The channel,

alternate channel, and unit specified must be declared on the system and must have a hardware type of DPK or DSK. Examples:

... CU=8/2 ...

If the MCP configuration file includes these records ...

DLP 7 DSK
DLP 8 DSK
EXCHANGE 8 7

....then the DISPKV input could be:

... CU 8:7/2 ...

AC RM
AC R
AC M
AC U

AC identifies the system access code parameter. This parameter is optional and specifies the diskpack type and access restriction. The available access codes and their names are:

RM (Restricted Master). Family names are required in order to access files on this type of diskpack subsystem. If you do not assign another name to this diskpack with the PN parameter, you must use the default family name SYS180 to access the files on this string of packs.)

R (Restricted). Assigns continuation diskpacks to a restricted master diskpack. The family names assigned to these continuation diskpacks must match the name of the string's restricted master diskpack.)

M (Master). Declares master diskpack for unrestricted system resource use. Family names are not required to access files on this type of diskpack subsystem.)

U (Unrestricted). This is the default. It declared continuation diskpacks to an unrestricted diskpack subsystem. Family names are not required to assess files on this diskpack.

Diskpacks with system access codes of M or RM are base diskpacks while those with codes of R or U are continuation diskpacks. Restricted diskpacks (system access code of R or RM) may contain only files having a multi-file-id identical to the diskpack name. The syntax is <family name>/<file-id>. For example, if a diskpack has a system access code of RM and a diskpack name, also called the family name, of FAM, all files written to that diskpack must be specified as FAM/<file-id>.

Unrestricted diskpacks (system access code of U or M) differ in that files can be assigned in either form - <pack-name>/<file-id> or <file-id>.

If AC is not included in the string, the default access code of unrestricted is used. If the system interchange code is set to IC, then the system access code must be unrestricted. Examples:

... AC=R ...

... AC RM ...

PN <pack name>

PN identifies the diskpack name parameter. The diskpack family name can be up to 17 characters

long but only the first six characters are recognized by the MCP. The name cannot be all blanks, nor can it contain special characters or embedded blanks. The default diskpack name is SYS180.

The diskpack name is also referred to as the family name for that diskpack. Files located on diskpack are referenced by multi-file-id consisting of <family name>/<file name>. For example, a file named BAND on a diskpack named ZEPHER would be referenced as ZEPHER/BAND.

Multiple diskpacks can have the same diskpack names but must have different serial numbers. One is designated as the base diskpack and the others as continuation diskpacks. The master diskpack has a system access code of RM (restricted master) or M (master). Continuation diskpacks have a system access code of R (restricted) or U (unrestricted).

A diskpack name of DISK, PACK, or DSKn (where n is any single character, including a space) is not allowed. These names are reserved for the MCP. Examples:

... PN=JAKE ...

... PN BACKUP ...

SC IC
SC MS

SC identifies the system interchange code parameter. This parameter is optional and specifies how the diskpack is to be used. When set to interchange (IC), the diskpack may be used to interchange files between various families of Unisys systems. When set to Medium System (MS), the diskpack is set up for use on Unisys V Series Systems.

The default setting for SC is MS.

Example:

... SC=MS ...

OI <owner's-id>

OI identifies the owner's-id parameter. The owner's-id is a field in the diskpack label. It can be up to 14 characters long with no embedded blanks. The default owner's-id is all blanks. Examples:

... OI=DOCGROUP ...

... OI ZEPHYRBAND ...

ER <error address>

ER identifies the error address parameter. This optional parameter is used to enter error addresses to DISPKV to identify, by address, possible bad spots on the disk media. DISPKV then relocates these addresses to spare sectors.

225-type diskpacks have a paper label attached which lists addresses that were found to be marginal by the factory Initialize, Verify, and Relocate (IVR) procedures.

When utilizing the DISPKV INITIALIZE function, the listed addresses are input using the form ER=CCCC/HH/SSS where:

CCCC = Cylinder number of the error address

HH = Head number of the error address

SSS = Sector number of the error address

A maximum of 50 addresses can be input. Examples:

... ER=24/7/65 ER 228/8/6 ...

... ER 12/3/31 ...

NOTE
Any bad areas on 235-type diskpacks have already been relocated and MUST NOT be input to DISPKV.

CY <cylinder number>

CY identifies the cylinder number parameter. This parameter is required for the SINGLE function and is optional for the 659IVR function. It specifies the cylinder or cylinders to be re-initialized. The cylinder number can be one to four digits in length and range between zero and the maximum number of cylinders on the diskpack. The maximum number of cylinders on each type and mode of diskpack appear in the following tables.

Type	Mode	Cylinders
225	In 100 or 180-byte mode	405
235	In 100 or 180-byte mode	811
206	Interlaced in 100 or 180-byte mode	814
206	Sequential in 100 or 180-byte mode	814
206	Binary Interlaced in 100 or 180-byte mode	814
206	Binary Sequential in 100 or 180-byte mode	814
207	Decimal Interlaced in 100 or 180-byte mode	1397
207	Decimal Sequential in 100 or 180-byte mode	1397
207	Binary Sequential in 100 or 180-byte mode	1563
207	Binary Interlaced in 100 or 180-byte mode	1563
659	Binary Sequential in 100 or 180-byte mode	2241
659	Binary Interlaced in 100 or 180-byte mode	2241
677	Binary Sequential in 100 or 180-byte mode	822
677	Binary Interlaced in 100 or 180-byte mode	822
680 *	Binary Sequential in 180-byte mode	884
682 *	Binary Sequential in 180-byte mode	1768
MD4 *	Storage Module Device (SMD) 180-byte mode	821
MD4 *	Storage Module Device (SMD) 100-byte mode	821
MD8 *	Extended Storage Module Device (XSMD) 180-byte mode	823

* The SINGLE function is not allowed with these drive types

Examples:

... CY=108 ...

... CY 228 ...

AD <address>

AD identifies the address parameter. This parameter is required for the RELOCATE function and specifies the decimal sector address to be relocated. The address can be one to six digits in length and range between zero and the highest addressable sector on the particular diskpack. The highest addressable sector on each type of diskpack is listed below:

Type	Mode	Address
225	In 100-byte mode	654974
225	In 180-byte mode	485169
235	In 100-byte mode	969144
235	In 180-byte mode	969144
206	Decimal Interlaced in 100-byte mode	565729
206	Decimal Sequential in 100-byte mode	476189
206	Decimal Interlaced in 180-byte mode	362229
206	Decimal Sequential in 180-byte mode	321529
206	Binary Interlaced in 100-byte mode	565729
206	Binary Sequential in 100-byte mode	476189
206	Binary Interlaced in 180-byte mode	362229
206	Binary Sequential in 180-byte mode	321529
207	Decimal Interlaced in 100-byte mode	998854
207	Decimal Sequential in 100-byte mode	998854
207	Decimal Interlaced in 180-byte mode	998854
207	Decimal Sequential in 180-byte mode	998854
207	Binary Interlaced in 100-byte mode	1117544
207	Binary Sequential in 100-byte mode	1117544
207	Binary Interlaced in 180-byte mode	1117544
207	Binary Sequential in 180-byte mode	1117544
659	Binary Interlaced in 100-byte mode	4696989
659	Binary Sequential in 100-byte mode	3957129
659	Binary Interlaced in 180-byte mode	3015489
659	Binary Sequential in 180-byte mode	2679189
677	Binary Interlaced in 100-byte mode	2185064
677	Binary Sequential in 100-byte mode	1841050
677	Binary Interlaced in 180-byte mode	1403214
677	Binary Sequential in 180-byte mode	1246844

680 *	Binary Sequential in 180-byte mode	2413320
682 *	Binary Sequential in 180-byte mode	4829369
MD4 *	Storage Module Device (SMD) in 180-byte mode	682259
MD4 *	Storage Module Device (SMD) in 100-byte mode	682259
MD8 *	Extended Storage Module Device (XSMD) in 180-byte mode	1389179

* The RELOCATE function is not allowed with these drive types

Examples:

... AD=8105 ...

... AD 680738 ...

FILES <number>

FILES identifies the files parameter. This parameter is optional and may only be used with the LABEL command. It specifies the number of files expected to be present on the diskpack.

As a result of using the files parameter, DISPKV will construct the appropriate number of contiguous directory sector entries for the specified number of files. Since the entire directory will be on a contiguous area of diskpack, head movement is minimized during directory lookup.

If the files parameter is not used, DISPKV assigns a default directory size large enough for 110 files. If the directory fills up, the MCP expands it by assigning additional areas, as available, elsewhere on the diskpack.

There is no two-letter mnemonic for the files parameter, the entire word must be used. Examples:

... FILES 3758 ...

... FILES 14284 ...

END "

END terminates ODT input. Accepts continue until this keyword is encountered. The quotation mark signals the end of the input data on this particular Accept statement. Examples:

... END"

12AX "END"

INITIALIZE COMMAND

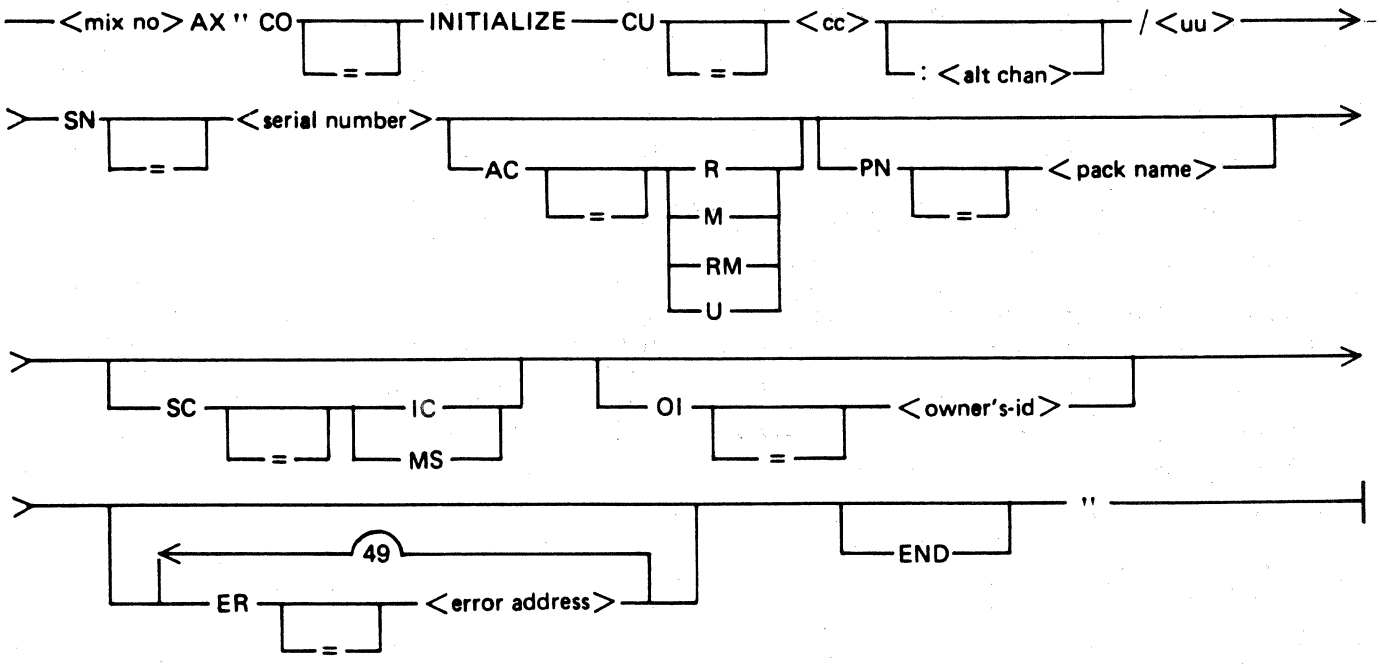


Figure 7-3. DISPKV INITIALIZE Syntax

INITIALIZE

The INITIALIZE command is the most comprehensive DISPKV command. It purges and completely recreates the entire diskpack. INITIALIZE performs three major functions - INITIALIZATION, VERIFICATION, and RELOCATION. Together these are abbreviated IVR.

The INITIALIZATION function directs the diskpack controller to write sector addresses and gaps in all sectors on the diskpack. Then a predefined data pattern is written in the data field of each sector.

The VERIFICATION function directs the controller to read and check all of the addressable sectors on the diskpack for address errors and protection code errors. In addition, the data field of each sector is compared against the predefined data pattern to check for validity.

The RELOCATION function specifies that if an error is detected during VERIFICATION, it is retried ten times. If the error is still present after the tenth retry, the sector in error is relocated to a spare sector. The relocate procedure writes the error sector's address into a spare sector address field and writes the predefined data pattern into the spare sector's data area.

After each relocate procedure, the verify procedure is done again to check for additional errors. A maximum of five relocates are allowed per cylinder. If errors occur after the fifth relocate, the rest of the cylinder, starting with the error address just received, will be removed from the master available table (XPed).

A maximum of 2000 XPs for each diskpack is allowed. After 2000 XPs, a message is displayed on the ODT stating the total number of errors and the cylinder number of the last error. The operator

then has the option to terminate DISPKV or to continue. If DISPKV is terminated, the total number of errors received for each read/write head will be displayed on the ODT. If the operator chooses to continue, DISPKV continues to remove sectors from the master available table.

When (IVR) is completed, DISPKV creates four items:

1. A diskpack label that contains the serial number, system access code, pack name, system interchange code, and owner's-id.
2. A master available table that contains all usable space on the diskpack.
3. An available table that contains all space not removed (XPed).
4. An empty diskpack directory table.

In addition, an error summary report is generated if any errors are encountered during the IVR process. This report contains any error result descriptors obtained and the I/O descriptor fired to the device.

Because of the error checking and the size of the diskpacks, the INITIALIZE function can take from 60 to 300 minutes. Processing time varies, depending on the job mix, channel activity, and the type of diskpack being initialized.

Required Parameters

There are three required parameters:

Command

Channel/unit

Serial number

Default values are used for all parameters not found in the input string. Detailed information regarding these parameters and their defaults can be found under Execution Syntax in this section.

Restrictions

The INITIALIZE function of DISPKV is not used to IVR (Initialize, Verify, and Relocate) 680, 682, MD4, and MD8 diskpacks. To IVR 680 or 682 diskpack, use the Peripheral Test Driver (PTD) program named M68IVR. To IVR MD4 or MD8 diskpack, use the PTD program named SMDIVR. The RECONFIGUREL function of DISPKV is then used to create the diskpack label, the available table and the diskpack directory table. PTD programs are available from your Unisys Customer Service Engineer (CSE).

The INITIALIZE function must be used when converting 100-byte diskpacks to 180-byte diskpacks and when converting 180-byte diskpacks to 100-byte diskpacks. It must also be used when converting interlaced to sequential mode or vice versa. When converting 207, 235, or 659 diskpacks, refer to the value statement in the execution syntax.

This function may or may not be valid for 659 diskpacks, depending on the situation.

- If you need to relocate as many as five sectors on each cyliner, you can use DISPKV to IVR a 659 diskpack.
- If you need to relocate more than five sectors, you must use a Peripheral Test Driver (PTD) program named SEQIVR (also known as the "shared spares" version of PTD IVR).

After the SEQIVR program has been used on a 659 diskpack, this function of DISPKV cannot be used on the diskpack.

Addresses are not written on 235 or 207 diskpack, but data fields are initialized.

The diskpack directory and available tables are created only on 180-byte diskpacks.

Examples:

```
8AX "CO INITIALIZE CU 8/2 SN 7142 AC=RM PN=MYPACK OI=MARIBEL"

8AX "CO INITIALIZE CU 6/4 SN 28391 END"
```

This syntax will cause DISPKV to IVR two diskpacks. The first one has a serial number of 7142 and is on unit 2 connected to DLP 8. It will be initialized as a restricted master with a diskpack name of MYPACK and an owner's-id of MARIBEL. The other parameters will assume default settings.

The second diskpack being IVRed has a serial number of 28391 and is on unit 4 connected to DLP 6. All optional parameters will assume default settings.

At the conclusion of the second INITIALIZE, DISPKV will go to end-of-job.

CONFIGURE COMMAND

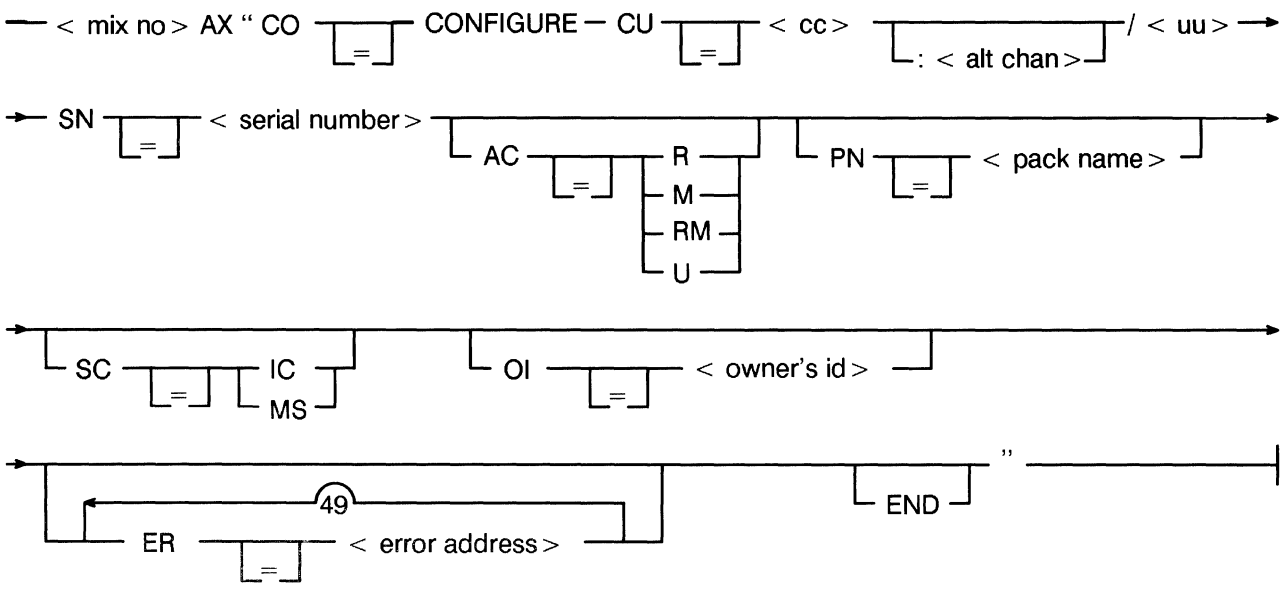


Figure 7.4 DISPKV CONFIGURE Syntax

CONFIGURE

The CONFIGURE command purges information on and performs VERIFICATION and RELOCATION functions on diskpacks that are either preinitialized from the factory or have been previously initialized with the DISPKV INITIALIZE command. The preinitialized diskpacks include the 207, 235, and 659 types.

The CONFIGURE command checks all addressable sectors on the diskpack for errors. Any sectors in error are then relocated to spare sectors. Any previously relocated sectors remain relocated. If the diskpack being checked is in 180-byte mode, the diskpack label, available table, and directory table are also created. If errors occur, an error summary report is generated. Table 7-2 provides a summary of DISPKV command functions.

Required Parameters

There are three required parameters:

Command

Channel/unit

Serial number

Default values are used for all parameters not found in the input string. Detailed information regarding these parameters and their defaults can be found under Execution Syntax in this section.

Restrictions

- The diskpack tables are created only on 180-byte diskpacks.
- This function is not valid for 680, 682, MD4, and MD8 diskpacks. To IVR 680 or 682 diskpack, use the Peripheral Test Driver (PTD) program named M68IVR. To IVR MD4 or MD8 diskpack, use the PTD program named SMDIVR. PTD programs are available from your Unisys Customer Service Engineer (CSE).
- This function may or may not be valid for 659 diskpacks, depending on the situation.
  - If you need to relocate as many as five sectors on each cylinder, you can use DISPKV on a 659 diskpack.
  - If you need to relocate more than five sectors, you must use a Peripheral Test Driver (PTD) program named SEQIVR (also known as the "shared spares" version of PTD IVR).
- After the SEQIVR program has been used on a 659 diskpack, this function of DISPKV cannot be used on the diskpack.

REPORT COMMAND

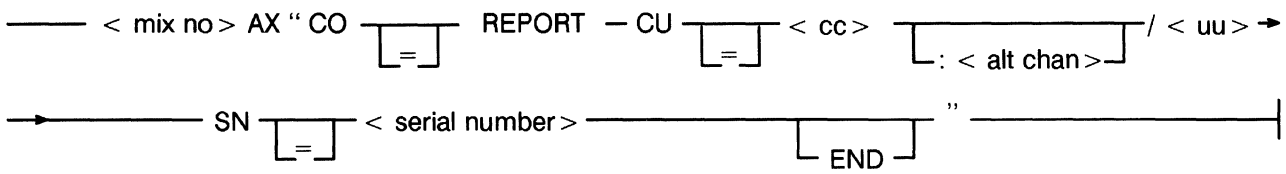


Figure 7-5. DISPKV REPORT Syntax

REPORT

The REPORT command scans all spare sectors on the diskpack and produces a printed report of all relocated sectors.

Required Parameters

There are three required parameters:

Command

Channel/unit

Serial number

All other parameters in the input string are checked for valid syntax but are not used. Detailed information regarding these parameters and their defaults can be found under Execution Syntax in this section.

Restrictions

This function is not valid for 680, 682, MD4, and MD8 type diskpacks.

This function may or may not be valid for 659 diskpacks, depending on the situation.

- If DISPKV has been used to IVR the 659 diskpack (relocating a maximum of five sectors on each cylinder), this function of DISPKV is valid.
- If the Peripheral Test Driver (PTD) program named SEQIVR (also known as the "shared spares" version of PTD IVR) has been used to relocate more than five sectors, this function of DISPKV is not valid.

SINGLE COMMAND

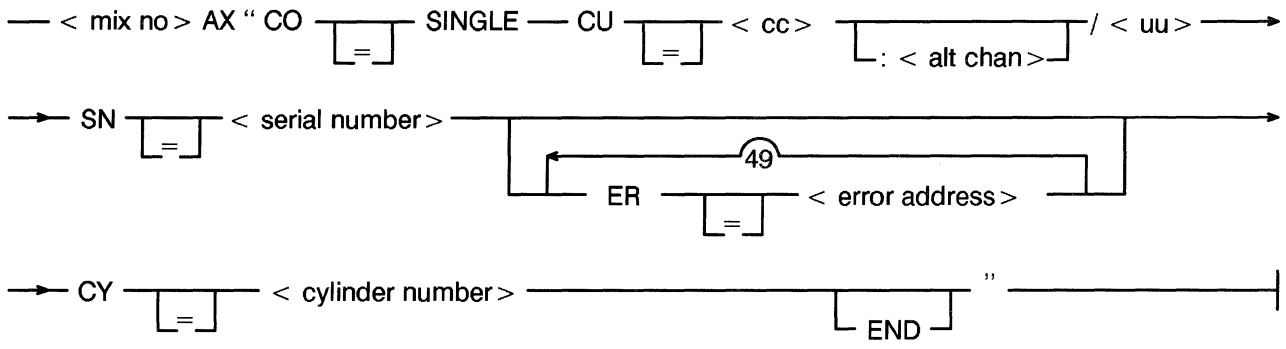


Figure 7-6. DISPKV SINGLE Syntax

SINGLE

The SINGLE command performs INITIALIZATION, VERIFICATION, and RELOCATION (IVR) functions on a single cylinder of a 100-byte or 180-byte diskpack. The diskpack label, tables, and other data on the diskpack will not be affected.

This procedure can be used if a particular cylinder on an initialized pack has an excessive number of errors, or if the addresses were somehow corrupted and a full diskpack IVR is not wanted. All addresses, error protection codes, and predefined data patterns are written on each sector of the specified cylinder. The sectors are then read to check for address errors, protection code errors, and data validity. Any bad sectors are relocated to spare sectors in the cylinder.

Before initializing and verifying a cylinder, DISPKV reads all the cylinder spares. If any of the spare sectors are in use, an ODT display identifies the relocated sectors and asks whether they are to be relocated after the cylinder is reinitialized. When the IVR is accomplished, any previously relocated sectors can be relocated again. Moreover, any ER addresses supplied during the process are also relocated.

Upon completing the entire SINGLE process, an ODT display appears showing all relocated sectors and other sectors that need to be removed.

Required Parameters

There are four required parameters:

Command

Channel/unit

Serial number

Cylinder

In addition, the address parameter can be used with the SINGLE command to identify addresses to be relocated. All other parameters in the input string are checked for valid syntax but are not used. Detailed information regarding these parameters and their defaults can be found under Execution Syntax in this section.

Restrictions

In order to perform a single cylinder IVR, the entire cylinder to be IVRed must be reserved using the XP system command.

This function of DISPKV is not valid for 680, 682, MD4, and MD8 diskpacks. To perform a single cylinder IVR on 680 or 682 diskpacks, use the Peripheral Test Driver (PTD) program named M68IVR. To perform a single cylinder IVR on MD4 or MD8 diskpacks, use the PTD program named SMDIVR. PTD programs are available from your Unisys Customer Service Engineer (CSE).

This function may or may not be valid for 659 diskpacks, depending on the situation.

- If you need to relocate as many as five sectors on a cylinder, you can use DISPKV to perform a single cylinder IVR on a 659 diskpack.
- If you need to relocate more than five sectors, you must use a Peripheral Test Driver (PTD) program named SEQIVR (also known as the "shared spares" version of PTD IVR).

After the SEQIVR program has been used on a 659 diskpack, this function of DISPKV cannot be used on the diskpack.

RELOCATE COMMAND

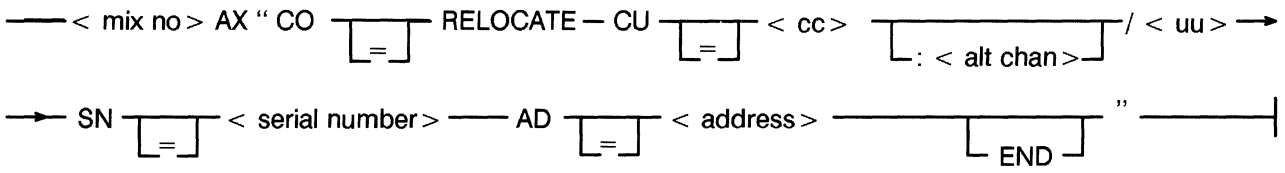


Figure 7-7. DISPKV RELOCATE Syntax

RELOCATE

The RELOCATE command takes the specified decimal address and relocates it into a spare sector on the cylinder if one is available. This procedure can be used when a sector on the diskpack produces an excessive number of errors and when a single cylinder or full diskpack Initialization, Verification, and Relocation (IVR) is not wanted.

If no spare sector is available, the request is aborted and a message is displayed on the ODT. After the sector is relocated, a verify pass is performed against the entire cylinder to check for errors in the relocation process. If the relocated sector is in error, the relocation is retried five times. After the fifth unsuccessful attempt, the operator is informed on the ODT and the process is terminated.

The diskpack label, tables, and all other data on the diskpack are not affected by the RELOCATE command.

Required Parameters

There are four required parameters:

Command

Channel/unit

Serial number

Address

All other parameters in the input string are checked for valid syntax but are not used.Detailed information regarding these parameters and their defaults can be found under Execution Syntax in this section.

Restrictions

The address to be relocated must be reserved using the XP system command.

This function of DISPKV is not valid for 680, 682, MD4, and MD8 diskpacks. To relocate an address on 680 or 682 diskpacks, use the Peripheral Test Driver (PTD) program named M68IVR. To relocate on MD4 or MD8 diskpacks, use the PTD program named SMDIVR. PTD programs are available from your Unisys Customer Service Engineer (CSE).

This function may or may not be valid for 659 diskpacks, depending on the situation.

- If DISPKV has been used to IVR the 659 diskpack (relocating a maximum of five sectors on each cylinder), this function of DISPKV is valid.
- If the Peripheral Test Driver (PTD) program named SEQIVR (also known as the "shared spares" version of PTD IVR) has been used to relocate more than five sectors, this function of DISPKV is not valid.

RENAME COMMAND

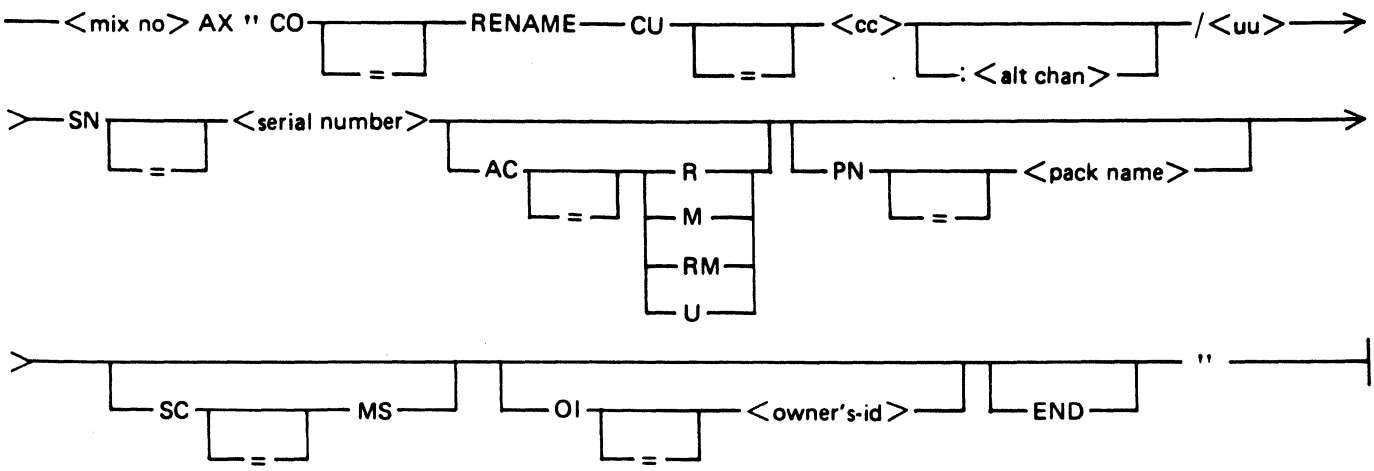


Figure 7-8. DISPKV RENAME Syntax

RENAME

The RENAME command changes various fields on the diskpack label without purging the diskpack or changing any tables. The following can be changed with the rename command:

Pack name

System access code

Owner's-id

Required Parameters

There are three required parameters:

Command

Channel/unit

Serial number

If diskpack name, system access code, and owner's-id appear in the input string, then the label is updated to these values. If they are not included in the input string, default values are assigned to these parameters.

All other parameters in the input string are checked for valid syntax but are not used. Detailed information regarding these parameters and their defaults can be found under Execution Syntax in this section.

Restrictions

The serial number must match the existing serial number in the label and the system interchange code, if included, must equal MS or the RENAME function aborts. This function is only valid for 180-byte diskpack.

RECONFIGURE COMMAND

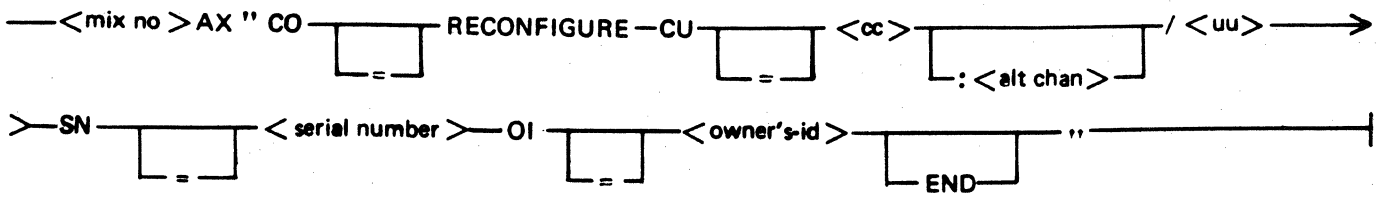


Figure 7-9. DISPKV RECONFIGURE Syntax

RECONFIGURE

The RECONFIGURE command purges all files on the diskpack and creates a new available table and empty directory using the existing master available table. The diskpack label is not changed.

Required Parameters

There are four required parameters:

Command

Channel/unit

Serial number

Owner's-id

All other parameters in the input string are checked for valid syntax but are not used. Detailed information regarding these parameters and their defaults can be found under Execution Syntax in this section.

Restrictions

The serial number and owner's-id parameters must match the existing serial number and owner's-id in the diskpack label or the RECONFIGURE function aborts. This function is only valid for 180-byte diskpack.

RECONFIGUREL COMMAND

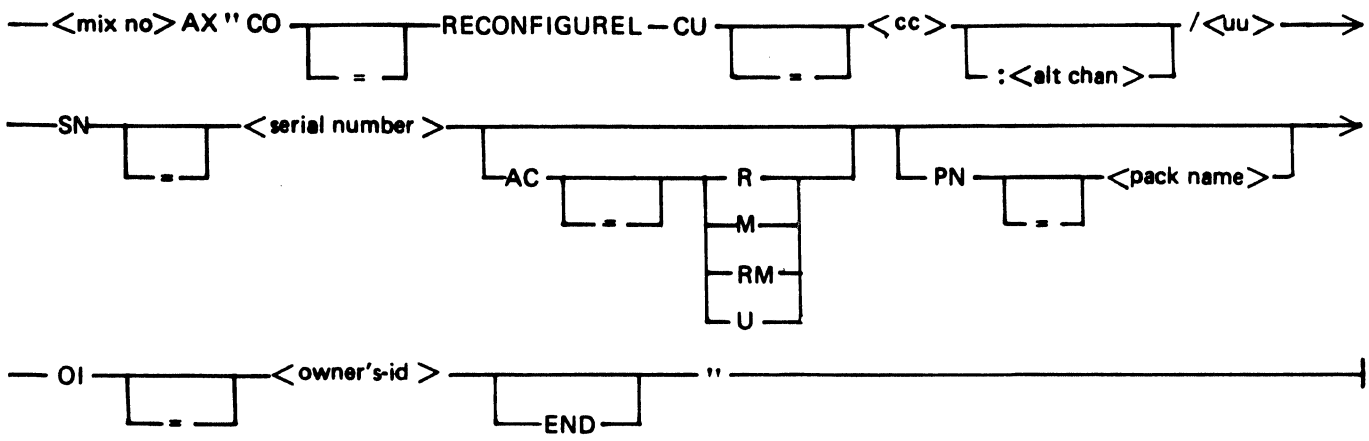


Figure 7-10. DISPKV RECONFIGUREL Syntax

RECONFIGUREL

The RECONFIGUREL command purges all files on the diskpack and creates a new available table and empty directory using the existing master available table. The diskpack name, system access code and owner's-id fields in the diskpack label can also be changed.

Required Parameters

There are four required parameters:

Command

Channel/unit

Serial number

Owner's-id

If diskpack name and system access code appear in the input string then the label is updated to these values. If they are not included in the input string, default values are assigned to these parameters. Detailed information regarding these parameters and their defaults can be found under Execution Syntax in this section.

Restrictions

This function is only valid for 180-byte diskpack.

LABEL COMMAND

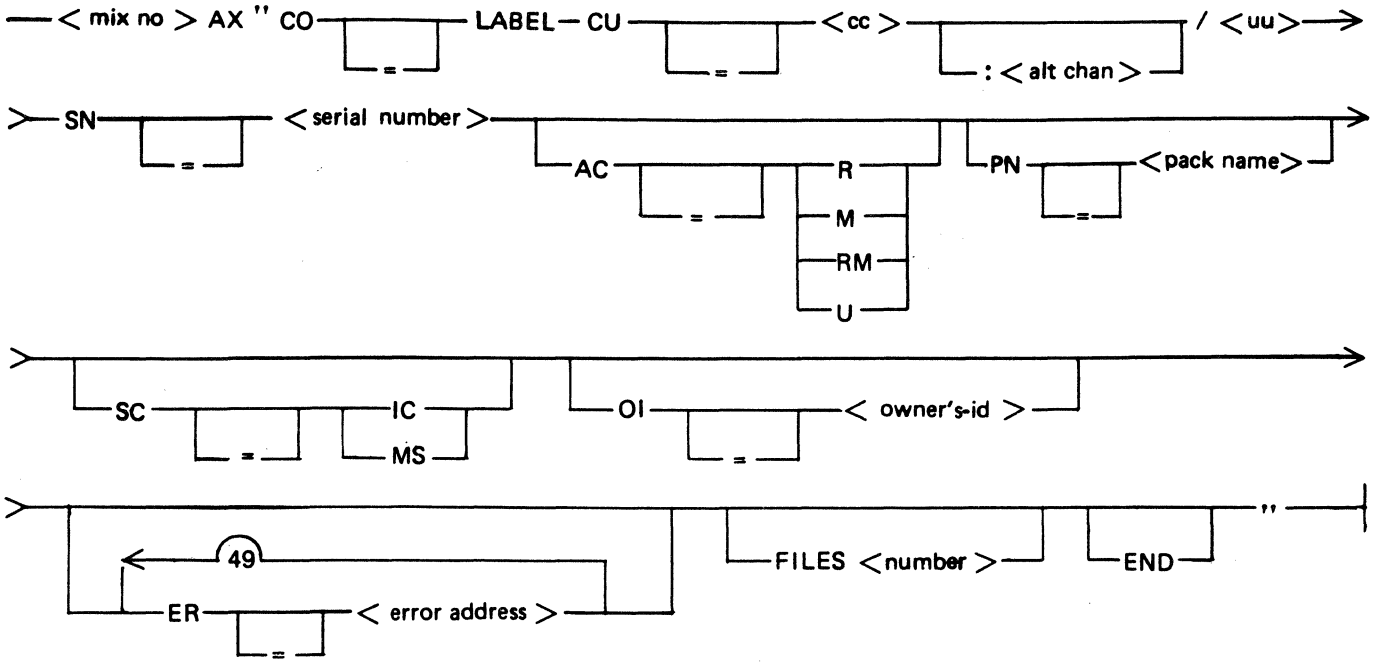


Figure 7-11. DISPKV LABEL Syntax

LABEL

The LABEL command purges all files on the diskpack and creates a new label, master available table, available table, and empty directory. The system access code, diskpack name, system interchange code and owner's-id fields in the diskpack label can be changed.

Required Parameters

There are three required parameters:

Command

Channel/unit

Serial number

If the system access code, diskpack name, system interchange code or owner's-id appear in the input string then a new label is created using these values. Default values are assigned to any of these parameters not included in the input string.

The files parameter may be used to establish a contiguous directory table large enough to hold the maximum number of files expected to be put on the diskpack. Detailed information regarding these parameters and their defaults can be found under Execution Syntax in this section.

Restrictions

This function is only valid for 180-byte diskpack. This function does not verify the diskpack and no error correction is performed. All previously unavailable (XPed) sectors will be ignored when building the new tables.

659IVR COMMAND

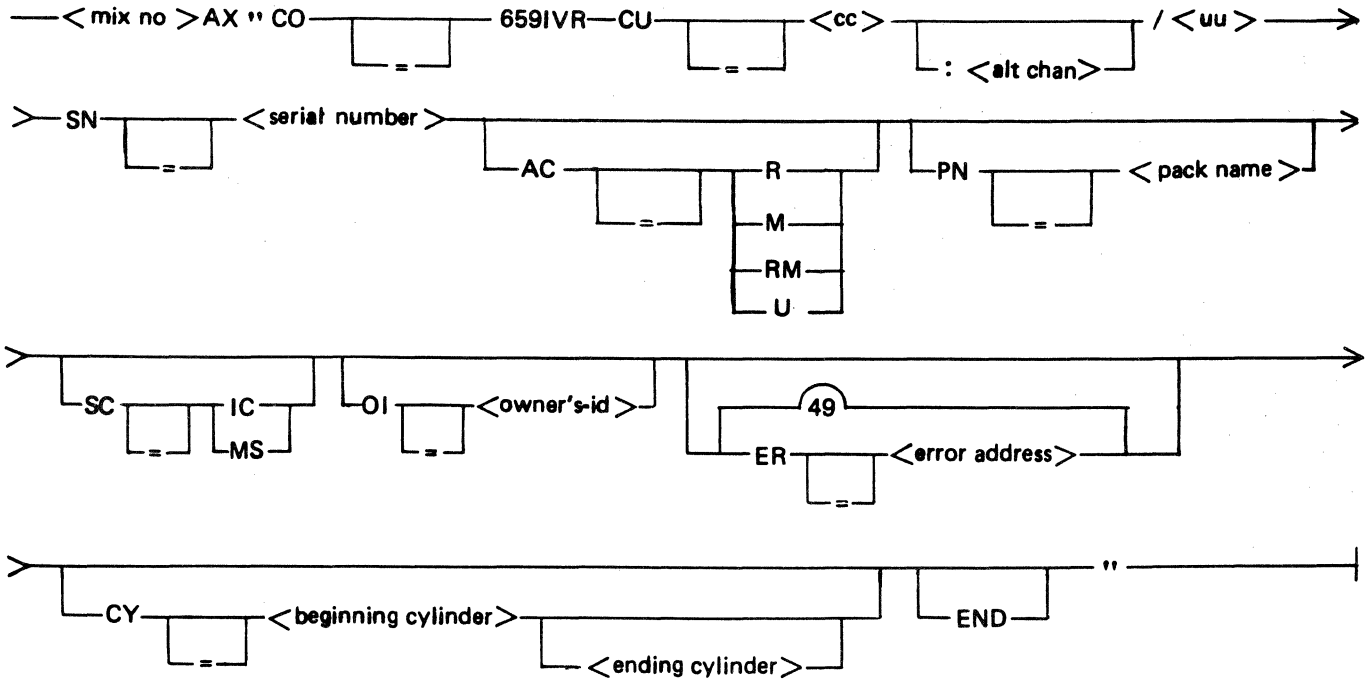


Figure 7-12. DISPKV 659IVR Syntax

659IVR

The 659IVR command performs an IVR on a single cylinder, on a selected range of cylinders, or on all cylinders on any type of diskpack except 680, 682, MD4, or MD8. The command unconditionally retains all previously relocated sectors and relocates any sectors newly found in error.

Depending on the range of cylinders chosen, the 659IVR function may or may not purge all files on the diskpack and create a new label, master available table, available table, and empty directory. The system access code, diskpack name, system interchange code and owner's-id fields in the diskpack label can be changed.

Required Parameters

There are three required parameters:

Command

Channel/unit

Serial number

If cylinder 0, which contains the label information, is to be initialized, then the system access code, diskpack name, system interchange code, and owner's-id should appear in the input string to be included in the new label. Default values are assigned to any of these parameters not included in the input string. Detailed information regarding these parameters and their defaults can be found under Execution Syntax in this section.

The <beginning cylinder> is the number of the cylinder to be initialized. If this parameter is not specified, then the entire diskpack is initialized. The <ending cylinder> is the number of the last cylinder to be initialized. This parameter is valid only if the <beginning cylinder> parameter has also been specified. If the <ending cylinder> parameter is omitted or if the value given is greater than the number of cylinders on the diskpack, then the initialization will be performed through the last cylinder. If <ending cylinder> is used, it must be separated from <beginning cylinder> by a dash.

If no cylinder numbers are specified, then DISPKV will ask the user to specify whether initialization is to begin with cylinder 0. If the response to this request is yes (Y), then DISPKV will IVR all cylinders on the diskpack. Otherwise, it will perform an IVR of cylinders one through the last cylinder on the diskpack.

If a range of cylinder numbers is specified beginning with cylinder 0, then DISPKV will ask the operator to confirm that cylinder 0 is indeed to be included. If the answer is no (N), the original request will be followed with the exception of cylinder 0.

Restrictions

This function is not valid for 680, 682, MD4, and MD8 type diskpacks. They are initialized using PTD programs.

Table 7-2. DISPKV Command Functions

	Init	Cnfig	Rpt	Sgl	Reloc	Rename	Recnf	Recnfl	Label	659IVR
Writes all sector addresses	Y	N	N	N	N	N	N	N	N	Y/N
Checks all addressable sectors	Y	Y	N	N	N	N	N	N	N	Y/N
Relocates sectors in error	Y	Y	N	N	N	N	N	N	N	Y
Purges all files	Y	Y	N	N	N	N	Y	Y	Y	Y/N
Creates new diskpack label	Y	Y	N	N	N	N	N	N	Y	Y/N
Updates existing diskpack label	N	N	N	N	N	Y	N	Y	N	N
Creates new master available table	Y	Y	N	N	N	N	N	N	Y	Y/N
Creates new available table	Y	Y	N	N	N	N	Y	Y	Y	Y/N
Creates new diskpack directory table	Y	Y	N	N	N	N	Y	Y	Y	Y/N
Creates error summary report if errors are found	Y	Y	N	N	N	N	N	N	N	Y
Creates report of all relocated sectors	N	N	Y	N	N	N	N	N	N	N
Writes sector addresses in a single cylinder	N	N	N	Y	N	N	N	N	N	Y/N
Checks addressable sectors in a single cylinder	N	N	N	Y	N	N	N	N	N	Y/N
Relocates sectors in error in a single cylinder	N	N	N	Y	N	N	N	N	N	Y/N
Relocates specified sectors	N	N	N	N	Y	N	N	N	N	N

RESPONSE TO ODT INPUT MESSAGES

If DISPKV asks for a specific input, for example, serial number or channel/unit, then only the data asked for should be entered. The keyword must not be re-entered. The keyword END should not be entered if specific input is asked for. If entered, it is ignored.

DISPLAYING STATUS INFORMATION

A switch can be set during execution which causes DISPKV to display status information. This is done using the SW system command to set switch 8 to one of three values.

SW8 may be used to interrogate DISPKV for the cylinder of the diskpack currently being initialized or configured, for the number of errors currently detected, or both. The syntax and results are shown below:

<mix no>SW8=1	Displays the current cylinder number.
<mix no>SW8=2	Displays the total number of errors encountered on a basis of each read/write head.
<mix no>SW8=3	Displays both the current cylinder number and the number of errors.

INITIALIZING MD4, MD8, 680, AND 682 DISKPACK

The DISPKV utility cannot be used to initialize MD4 (SMD), MD8 (XSMD), 680, or 682 diskpack types. The PTDMNO program, supplied on the MCP system tape, must be used.

Before attempting to initialize media with the PTDMNO program, the PTDMNI, M68FVI, and SMDIVI files should be loaded to disk from the MCP system tape and printed. These files contain documentation on the operation of the PTDMNO program and are in printer backup disk format.

DISPKV can be used to rename and relabel MD4, MD8, 680, and 682 diskpack.

DISKPACK AND CONTROLLER STYLES SUPPORTED

DISPKV supports many different models of diskpack controllers and diskpacks.

The controllers supported are:

BX383 DPDC through an HT-DLP

B X384 DPDC through an HT-DLP

B X385 DPDC through an HT-DLP

B 9387 DPDC through an HT-DLP, or SEQ-HT-DLP

B 9389 DPDC through an SEQ-HT-DLP

NOTE

SMD and XSMD disk do not use a diskpack controller. They are connected directly to a DLP.

The diskpacks supported are shown in Table 7-3.

Table 7-3. Available Disk and Diskpack Styles

Disk—100-Byte

Style	Type	Notes
B 9486-4	225	Dual spindle, removable
B 9484-8	235	Dual spindle, removable
B 9484-5	206	Dual spindle, removable
B 9494-41	207	Dual spindle, fixed
B 9484-12	677	Memorex removable
B 9494-5	659	Memorex fixed
MD4-2 or -4	MD4	SMD fixed

Diskpack—180-Byte

Style	Type	Notes
B 9486-4	225	Dual spindle, removable
B 9484-8	235	Dual spindle, removable
B 9484-5	206	Dual spindle, removable
B 9494-41	207	Dual spindle, fixed
B 9484-12	677	Memorex removable
B 9494-5	659	Memorex fixed
B 9494-12	680	Memorex fixed, dual actuator
B 9494-24	682	Memorex fixed, dual actuator—double density
MD4-2 or -4	MD4	SMD fixed
MD8-2 or -4	MD8	XSMD fixed—double density

The DISPKV commands that are valid for each diskpack type are shown in Table 7-4.

Table 7-4. DISPKV Commands and Qualified Disk Types

Disk—100-Byte Media

	Init	Config	Report	Sgl	Relo-cate	Rename	Recon-fig	Recon-figl	Label	659IVR
225	yes	yes	yes	yes	yes	N/A	N/A	N/A	N/A	yes
235	yes	yes	yes	yes	yes	N/A	N/A	N/A	N/A	yes
206	yes	yes	yes	yes	yes	N/A	N/A	N/A	N/A	yes
207	yes	yes	yes	yes	yes	N/A	N/A	N/A	N/A	yes
677	yes	yes	yes	yes	yes	N/A	N/A	N/A	N/A	yes
659	yes	yes	yes	yes	yes	N/A	N/A	N/A	N/A	yes
MD4	no	no	no	no	no	N/A	N/A	N/A	N/A	no

Diskpack—180-byte Media

	Init	Config	Report	Sgl	Relo-cate	Rename	Recon-fig	Recon-figl	Label	659IVR
225	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes
235	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes
206	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes
207	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes
677	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes
659	yes *	yes *	yes *	yes *	yes *	yes	yes	yes	yes	yes
680	no	no	no	no	no	yes	yes	yes	yes	no
682	no	no	no	no	no	yes	yes	yes	yes	no
MD4	no	no	no	no	no	yes	yes	yes	yes	no
MD8	no	no	no	no	no	yes	yes	yes	yes	no

Rules Regarding 659 Diskpacks

*

These five functions may or may not be valid for 659 diskpacks, depending on the situation. If DISPKV has been used to IVR the 659 diskpack (relocating a maximum of five sectors on each cylinder), these functions of DISPKV are valid. If the Peripheral Test Driver (PTD) program named SEQIVR (also known as the "shared spares" version of PTD IVR) has been used to relocate more than five sectors, this function of DISPKV is not valid.

FILE EQUATE INFORMATION

In executing DISPKV, it may be necessary in a multi-programming environment to know the internal and external names used by DISPKV for all its input and output files.

DISPKV input card file

Internal name INPUTF external name INPUTF.

DISPKV output error summary report

Internal name PRINTF external name PRINTF.

DISPKV output DLP error report

Internal name DLPERR external name DLPERR.

INPUT EXAMPLES

Both ODT and card input are allowed. Valid ODT input is shown below. Valid card input is shown in figure 7-13.

Valid ODT inputs are:

EX DISPKV (,"SPO")

Enter input for DISPKV. The entire line must be enclosed with quotation marks. END stops Accepts.

**DISPKV=001 ACCEPT

1 AX "CO CONFIGURE CU 8/1 SN 2 CO INITIALIZE"

**DISPKV=001 ACCEPT

1 AX "CU 8:7/5 SN 11 AC RM ER 1/1/1 ER 2/3/4"

**DISPKV=001 ACCEPT

1 AX "PN JAKES OI IVRSUN"

**DISPKV=001 ACCEPT

1 AX "CO SINGLE CU 8/4 SN 875 CY 300 ER 300/6/29"

**DISPKV=001 ACCEPT

1 AX "CO RELOCATE CU 8/6 SN 39 AD 385472"

**DISPKV=001 ACCEPT

1 AX "CO CONFIGURE CU 6:5/4 END"

RESULTS OF INPUT

Assuming that each of the units designated is a 180-byte diskpack, the results of the above examples are as follows.

Input - "CO CONFIGURE CU 8/1 SN 2 CO INITIALIZE"

Unit 8/1 is verified, all files purged, diskpack label written, and pack directory and tables initialized. The diskpack serial number is 2. The pack is a system resource (unrestricted) diskpack, with a default diskpack name of SYS180, a system interchange code of MS (Medium System), and an owner's-id of all blanks. An INITIALIZE is done on the diskpack specified by the next CU that is input to DISPKV.

Input - "CU 8:7/5 SN 11 AC RM ER 1/1/1 ER 2/3/4 PN JAKES OI IVRSUN"

Unit 5 is INITIALIZEd using DLP 7 (the alternate channel) rather than 8 (the primary channel) as a restricted master with a serial number of 11, a diskpack name of JAKES, an owner's-id of IVRSUN, and a system interchange code of MS (Medium System). The ER 1/1/1 specifies that cylinder 1, head 1, sector 1 is to be relocated. The ER 2/3/4 specifies that cylinder 2, head 3, sector 4 is to be relocated.

Input - "CO SINGLE CU 8/4 SN 875 CY 300 ER 300/6/29"

A single cylinder IVR is done to cylinder 300 of the diskpack on channel 8, unit 4. The serial number of the diskpack must be 875. The 29th sector, as accessed by head 6, is relocated.

Input - "CO RELOCATE CU 8/6 SN 39 AD 385472"

A single sector RELOCATE of sector 385472 is done on channel 8, unit 6. The diskpack on channel 8 unit 6 must have a SERIAL NUMBER of 39.

Input - "CO CONFIGURE CU 6:5/4 END"

Unit 4 is initialized using channel 5 (the alternate channel) rather than channel 6 (the primary channel), verified, all files purged, diskpack label written, and diskpack directory and tables initialized. After the configure is complete, DISPKV goes to EOJ.

CARD INPUT RULES

As with ODT input, order when inputting DISPKV parameters on cards is unimportant except that the CO is used to group a set of data items. Everything following the keyword CO is assumed to belong to the same set until the next keyword CO is encountered.

The input data can be placed on several cards and can start in any column. However, the keyword must NOT cross card boundaries. The data associated with a keyword also must NOT cross card boundaries. As with ODT input, the keyword can be followed by an equal sign (=), for instance, CO=INITIALIZE CU=8/3 SN=11 PN=MASTER. If the keyword is not followed by an equal sign, it must be followed by at least one blank. A sample DISPKV execution and parameter deck is shown in Figure 7-13.

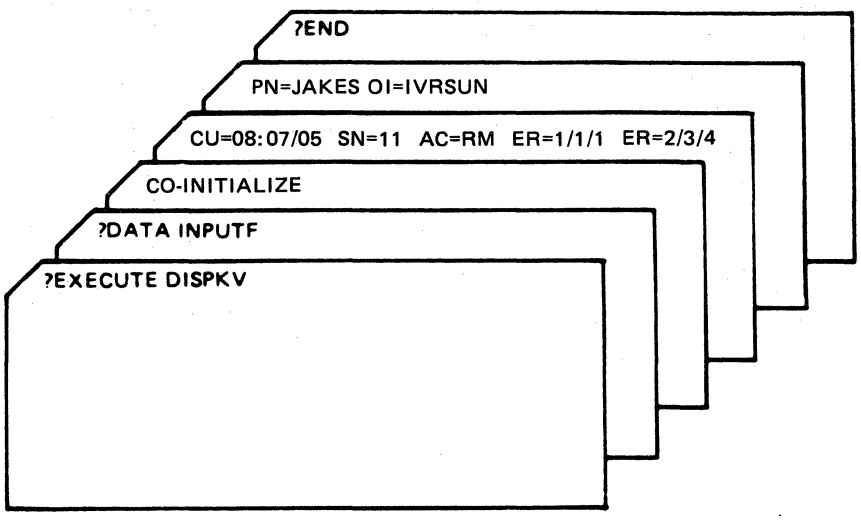


Figure 7-13. Sample DISPKV Execute Deck

DISPKV MESSAGES

The messages displayed by DISPKV fall into these three categories:

- Informational messages - These messages indicate the completion of a function and require no operator intervention.
- Operator prompts - These messages prompt you for some type of input.
- Error messages - These messages indicate problems that occurred when running DISPKV.

Informational Messages

The following are informational in nature. No response is required. Each message is followed by a description.

The following message is displayed on the ODT after a single cylinder IVR function if the relocation of a sector was required or if a portion of the cylinder must be removed from the disk available table (with XP system command for 180-byte diskpack or XD system command for 100-byte diskpack).

CYLINDER <number> ON CC/U HAS BEEN INITIALIZED AND

THE FOLLOWING ADDRESSES WERE RELOCATED

1 = ccc/hh/ssssss (<sector>) 2 = ccc/hh/ssssss (<sector>)

3 = ccc/hh/ssssss (<sector>) 4 = ccc/hh/ssssss (<sector>)

5 = ccc/hh/ssssss (<sector>)

ALSO, THE FOLLOWING ADDRESSES MUST BE XP'ED

ccc/hh/ssssss THRU ccc/hh/ssssss (<sector> FOR <length>)

A sector was corrupted and could not be relocated. The rest of the cylinder could not be verified and therefore it must be removed from the available table. On 100-byte diskpacks, remove the addresses

specified on the ID number to which the diskpack is assigned. On 180-byte diskpacks, replace the addresses of the cylinder that were removed prior to executing DISPKV (using the RXP system command), and then remove the addresses in the display. An example of this display is:

```
CYLINDER 10 ON 14/06 HAS BEEN INITIALIZED AND

THE FOLLOWING ADDRESSES WERE RELOCATED

1 =10/1/60 (4600) 2 =10/1/62 (4602)
3 =10/1/63 (4603) 4 =10/1/82 (4622)
5 =10/4/22 (4832)

ALSO, THE FOLLOWING ADDRESSES MUST BE XP'ED

10/4/33 THRU 10/4/84 (4843 FOR 52)

PACK TYPE 207: DATA FIELD ONLY INITIALIZE
```

A 207 diskpack is being initialized. Only the data portion of every sector is initialized; the addresses are not affected. If any addresses are corrupted, a single cylinder IVR must be performed on the associated cylinder or setting VA 1 = 100000 will override this if a full IVR is needed.

```
PACK TYPE 235: DATA FIELD ONLY INITIALIZE
```

A 235 diskpack is being initialized. Only the data portion of every sector is initialized; the addresses are not affected. If any addresses are corrupted, a single cylinder IVR must be performed on the associated cylinder.

Operator Prompts

When DISPKV displays the following prompts, you must provide a response before processing can continue.

This message will appear when doing a single cylinder IVR if any of the spares on that cylinder are in use.

```
WANT TO RELOCATE THESE ADDRESSES ON CC/U AGAIN?

1 = cccc/hh/ssssss (<sector>) 2 = cccc/hh/ssssss (<sector>)
3 = cccc/hh/ssssss (<sector>) 4 = cccc/hh/ssssss (<sector>)
5 = cccc/hh/ssssss (<sector>)

ENTER YES, NO, OR # OF ONES TO RELOCATE
```

If all of the sectors are to be relocated again after the initialize, enter YES. If none are to be relocated again, enter NO. If selective ones are to be re-relocated, enter the number that precedes the = sign that corresponds to the address to be relocated. For example:

```
WANT TO RELOCATE THESE ADDRESSES ON 14/6 AGAIN?

1 = 10/4/22 (4832) 2 = 10/1/60 (4600)
3 = 10/1/62 (4602) 4 = 10/1/63 (4603)
5 = 10/1/82 (4622)
ENTER YES, NO, OR # OF ONES TO RELOCATE
```

All of the sectors displayed were relocated before the single cylinder IVR was done. Any or all of the sectors may be relocated again, if desired. Entering a YES will re-relocate all five sectors. Entering a NO will result in none of the five sectors being re-relocated. Entering the numbers 234 will result in sectors 4600, 4602, and 4603 being relocated again.

Error Messages

The following are various error messages sent by the system to the ODT.

ACCESS CODE MUST BE BLANK FOR INTERCHANGE PACKS

The system interchange code specified is IC and the system access code is not unrestricted. The system access code must be unrestricted on all interchange diskpacks.

COMMAND IS NOT ALLOWED TO THIS UNIT

This message will be displayed when any of the following commands is entered for a 680, 682, MD4, or MD8 diskpack: INITIALIZE, CONFIGURE, SINGLE, REPORT, or RELOCATE.

CONTROLLER ERR - RD TAG PRESENT

A DLP extended result descriptor R/D tag was returned specifying a controller malfunction, invalid disk Device Dependent Port (DDP) status, controller failure, and so forth.

DLP ERROR R/D RETURNED ON THE DLP

Invalid result descriptor was returned from the DLP. Possible causes are DLP descriptor error, parity error, Device Dependent Port (DDP) parity error, timeout, and so forth.

ERROR CARDS IGNORED, COMMAND IS NOT INITIALIZE

Error addresses can be specified only when using the initialization function.

ERRORS PER HEAD: <head number>=<error count> <head number>=<error count>

This shows the total number of errors encountered for each read/write head. This message appears only when the total count exceeds 2000 and the operator terminates DISPKV.

IOT ERROR CONDITION ON THE DLP

An invalid result descriptor was returned from the IOT. Possible causes are timeout, memory parity error, longitudinal parity error, and so forth.

INVALID INPUT CYLINDER NUMBER

The cylinder number input to the SINGLE function exceeds the maximum number of cylinders for that diskpack.

INVALID LABEL ON MEDIA DESIGNATED

Label on the channel and unit specified is in the incorrect format or cannot be read.

INVALID OWNER'S-ID FOR RECONFIGURE

PACK OWNER'S-ID on the channel and unit specified does not match the OWNER'S-ID input to DISPKV. RECONFIGURE must have matching OWNER'S-IDs.

INVALID PACK SERIAL NUMBER FOR RECONFIGURE

PACK SERIAL NUMBER on the channel and unit specified does not match the serial number input to DISPKV. RECONFIGURE must have matching serial numbers.

INVALID RELOCATE ADDRESS

The sector address input to the RELOCATE function exceeds the maximum addressable sector for that type of diskpack.

INVALID R/D ON READ = <result descriptor>

An attempt to read the spare sectors of a cylinder has failed. The result descriptor specified was returned from this read.

INVALID R/D RECEIVED ON ERROR RELOCATE = <result descriptor>

Invalid result descriptor was returned on an error sector relocate. Possible causes are parity error, write fault, address error, and so forth.

INVALID R/D RECEIVED ON I/O <result descriptor>
R = RETRY, N = NEXT COMMAND STRING
D = DUMP, NEXT COMMAND STRING

An error was found when doing an I/O operation. A response of "R" will cause the I/O to be retried ten times or until it is successful. If the I/O fails again after ten retries the same message is displayed. A response of "N" will cause the current command to be terminated and DISPKV will process the next command string. If no further commands are available, DISPKV terminates. A response of "D" is the same as "N" except that a memory dump will be taken.

INVALID R/D RETURNED ON READ UNIT = <result descriptor>

An invalid result descriptor was returned from the controller when attempting to obtain the firmware revision level and drive type. Possible causes are that the controller is off-line, the unit is in maintenance mode, the unit is not present, and so forth.

LABEL COULD NOT BE READ

The label sector cannot be read after 10 retries.

PACK TYPE (IE 235, 206) IS UNDEFINED

Pack type returned on read unit-id is not supported by this version of DISPKV. The cause is incorrect firmware loaded into the controller or an incorrect version of DISPKV.

PACK TYPE RETURNED ON TEST OP IS INVALID

The diskpack type returned on read unit-id is not supported by this version of DISPKV. The cause is incorrect firmware loaded into the controller or an incorrect version of DISPKV.

REQUEST NOT ALLOWED, ADDRESS <number> FOR <length> NOT XP'ED

The address and length specified must be removed before doing a single cylinder IVR (SINGLE). Remove these addresses using the XP system command on 180-byte diskpacks or the XD system command on 100-byte diskpacks, and re-execute DISPKV.

REQUEST NOT ALLOWED, ADDRESS <number> NOT XP'ED

The address specified in a RELOCATE function for a 180-byte diskpack was not removed from the disk available table. Remove the appropriate address using the XP system command and re-execute DISPKV.

REQUEST NOT ALLOWED ON 100-byte MEDIA

Command of RECONFIGURE or RECONFIGUREL is not allowed on 100-byte diskpacks. Tables are not used on these diskpacks.

SECTOR <number> ON <cc/u> CANNOT BE RELOCATED, LEAVE IT XP'ED

The sector input to DISPKV cannot be relocated because no spares are available for use, or the available spares are bad. This sector should be removed from the available table (XPed) until a full diskpack IVR or a single cylinder IVR can be performed.

SECTOR <number> ON <cc/u> CANNOT BE VERIFIED, <number> IS BAD

An attempt to verify the cylinder containing the sector address to relocate failed. The sector specified did not pass the verify and should also be relocated. The original sector input to DISPKV was relocated.

INITIALIZING DISKPACKS TO 100-BYTE MODE

The following information provides procedural guidelines for diskpack preparation in 100-byte mode.

Perform the following steps.

1. Ensure that the drive has been strapped for 100-byte mode.
2. Declare the spindles using the DISK MCP configuration record. Use the SAVED or RESERVED option so the MCP will not attempt to use the drive for storage. See the *V Series MCP/VS System Software Operations Guide, Volume 1: Installation.*
3. Execute DISPKV and initialize the media.
4. Ready the drive (RY system command).

SECTION 8
PSEUDO READER LOAD CONTROL PROGRAM (LDCNTL)

The Pseudo Reader Load Control Program (LDCNTL) lets you create a disk or diskpack file, called a pseudo card file, from a card deck. A pseudo card file is also referred to as a control deck, and may contain data, MCP commands, or MCP commands and data. Although LDCNTL is a utility bound to the MCP, you can code and use your own version of the utility if you wish.

Creating a pseudo card file from a card deck is useful for these functions:

- Compiling programs that are punched on cards.
- Running programs or program systems by punching the necessary commands on cards, and then reading the cards.
- Carrying out commands that are punched on cards.
- Reading data that is punched on cards.
- Improving the efficiency of a card-based system.

Once the deck has been input and you have created the pseudo card file, you must activate it before you can run it. The activation process depends on the procedure you use to create a pseudo card file. If you use the manual procedure, you must use the RN or RNP command to activate the pseudo card file. If, on the other hand, you set the USE APCR option, the MCP automatically activates the pseudo card file for you.

If you use cards, you can improve the efficiency of your system with pseudo card files in the following ways:

- Programs can read several card files simultaneously, even though you have only one or two card readers.
- Reading data in the form of a pseudo card file is faster than reading data in from a card reader.
- You can use LDCNTL to transfer your files to disk or diskpack devices and make the files accessible to many jobs simultaneously.

If you will need to use the pseudo card file more than once, follow the manual procedure to create your pseudo card file. Otherwise, the pseudo card file will be purged from disk or diskpack after it is used. Refer to the Operating Procedures portion of this section for detailed procedures.

LDCNTL SYSTEM COMMANDS AND OPTIONS

When you use the LDCNTL program, you may use a number of system commands (including MCP control instructions and keyboard input messages) and MCP options. Specifically, you will use system commands to direct the MCP to perform particular actions. There are two types of system commands: MCP control instructions and keyboard input messages.

You can enter MCP control instructions on cards or from the keyboard. You can enter keyboard input messages from the keyboard only. The *MCP/VS System Software Operations Guide Volume 2: System Commands* provides more detailed information about each of these command types, including the specific commands.

Below is a listing of the MCP control instructions and system commands you can use with the LDCNTL program:

LDCNTL MCP Control Instructions

The following control instructions pertain to LDCNTL:

- AX
- CHANGE
- CHARGE
- COMPILE
- DATA/DATAB (DATA CTLDCK)
- END/ENDCTL
- EXECUTE (LDCNTL)

LDCNTL Keyboard Input Messages

The following keyboard input messages pertain to LDCNTL:

- AX
- CD, CDP
- CV, CVP
- DA, DAP
- DQ
- GO
- LD, LDP
- OL
- RD, RDP
- RN, RNP
- RO
- SD
- SO
- TO

NOTE
The AX command is both an MCP control instruction and a keyboard input message.

The MCP provides a number of options. You can use a USE record to specify that a certain option should be used. If you omit the USE record, the option will not be used. You can use the SO; RO, TO commands to set, reset, or interrogate an option.

The *V Series MCP/VS System Software Operations Guide Volume 1: Installation* provides more detailed information about the following USE records that affect the treatment of pseudo card files:

- APCR
- AURD
- PCRM

LIMIT Records set the sizes of certain MCP tables and the values of certain system parameters. Refer to the *V Series MCP/VS System Software Operations Guide Volume 1: Installation* for information about the limit on pseudo card files.

When Not to Use Pseudo Card Files

You should use card readers instead of pseudo card files if any of these conditions apply:

- You only use card decks occasionally.
- You do *not* do a lot of multiprogramming work that involves cards. Multiprogramming means that you have more than one program running at the same time.
- Only a few of your programs use card files.

Other considerations include:

- The amount of time it takes to load a card file to disk or diskpack.
- The amount of space a large card deck requires on the disk or diskpack.
- The small amount of memory needed for the pseudo card file processing.
- The requirement that you have punched card equipment.

Creating a Psuedo Card File

You may use any of the following methods to create a pseudo card file:

- Enter an LD or LDP command or an EXECUTE LDCNTL command.
- Enter the command CHANGE <file name> TO #. This command assigns a pseudo card file number to a disk or diskpack file; however, the file must already be in pseudo card file format. The MCP will assign a number to the pseudo card file (see File Names). Write this number down so you can activate the pseudo card file later with the command RN or RNP. The following message will appear on your terminal:<file name> CHANGED TO #pnnnn
- Use the CANDE (Command and Edit System) to enter a COPY TO CTLD command. Refer to the *V Series CANDE Installation and Operations Guide* for more information.
- Write a user program that creates files in pseudo card file format (refer to Files and Records for format requirements).

When you create a pseudo card file, the output disk or diskpack file identifier must be #00000, even if you create a pseudo card file with a user program independently of LDCNTL. #00000 is a special identifier that tells the MCP to assign the next sequential pseudo card file number when a file is opened.

File Names

All pseudo card files have a name of the form: #pnnnn.

- The pound sign indicates this as a pseudo card file.
- The letter p represents the number of the processor on which the file was created.
- The letters nnnn represent the number that the MCP assigns the file.

Requirements for a Control Deck

When you are going to read in a card deck from a physical card reader, you must first include two special cards that indicate where the deck begins and ends.

The first card in your card deck must be *?DATA CTLDCK*. The last card tells the MCP that you want to end the LDCNTL program. Therefore, the *last card* in your *last card deck* must be *?ENDCTL*.

The MCP only processes SPO control instructions as it creates the pseudo card file. SPO control instructions enable you to enter keyboard input messages from a card reader. The cards must be punched in the following format: ?SPO <text> where text is the keyboard input message.

The MCP reads all other control instructions, including the ?END cards, as data. It does not execute these commands unless the pseudo card file is automatically activated, or manually activated and executed.

In the CTLDCK file, the deck includes all physical cards from the ?DATA CTLDCK card through the ?ENDCTL card. The data file includes all cards between ?DATA CTLDCK and ?ENDCTL (except SPO control instructions), even though some contain invalid characters in column 1.

If you set the MCP option AURD, pseudo card files with erroneous combinations of control instructions will be removed automatically. If you do not set the AURD option, the MCP will display the message CONTROL CARD ERROR followed by an ACCEPT message. You must then rectify the erroneous condition before you can use an AX command to tell LDCNTL to continue. An example of an erroneous combination of control instructions is an ?END card that is not followed by a control instruction or the special ?ENDCTL card.

Initiating Load Control

You can initiate LDCNTL at any time as described in Creating A Pseudo Card File. The command LDP puts the pseudo card file on diskpack. The command LD puts the pseudo card file on disk, one file per card reader.

If you execute LDCNTL when you first initialize the system, you can leave it in the mix until processing is complete for the day. In this case, you would not use the ?ENDCTL card until you had read in the last card deck for the day. If LDCNTL does remain in the mix all day, it will be idle most of the time. However, this does not consume much in the way of system resources.

You can enter more than one LD or LDP command at one time. However, if you input more LDCNTL messages than there are card readers, you will not increase productivity, and you will use more memory.

The MCP does not require that the LDCNTL program be in the mix to process pseudo card files.

The cards that you read in can be in EBCDIC format, BCL format, or both. However, you must put a ?DATA control instruction in front of those cards that are in EBCDIC format, or a ?DATAB control instruction in front of those cards in BCL format. LDCNTL cannot process *BINARY* decks.

If the input from which a pseudo card file is made comes from a Remote Job Entry (RJE) terminal, the program should execute a PROGRAM BCT to obtain its RJE link. Place the RJE link value in columns 75-76 of the control instructions in the pseudo card file, a move of 2 UA to 2 UA, and put @FF@ in column 74. This link value provides a route back to the RJE station for messages that the pseudo card file generates.

MCP Capabilities in LDCNTL

A pseudo card file can contain as many as 100,000 card images. When you activate a pseudo card file, the MCP creates a record in the disk resident Pseudo-Reader Cross Reference Directory (PCRXRF). The PCRXRF is capable of handling 80 active pseudo card files for each processor. A total of 9999 pseudo card files may be present on disk for each processor; however, only 80 can be active simultaneously.

With the SD command, you can further restrict the number of pseudo card files the APCR facility activates simultaneously. Enter SD <integer> where the integer is any value from 1 to 80.

If the directory becomes full or the system reaches the limit, the MCP temporarily stops activating pseudo card files. You can override this pause with the RN or RNP command, which will activate more pseudo card files up to a maximum of 80 for each processor.

OPERATING PROCEDURES

If you want to use a pseudo card file without reading it in a second time, you must use the manual procedure. If you intend on using it only once, you can use the automatic procedure, which will then remove the pseudo card file automatically after processing completes.

The following steps outline a procedure for creating pseudo card files. Steps 1-6 and step 12 apply to both procedures.

Setting Up

1. Punch the cards ?DATA CTLDCK and ?ENDCTL.
2. Place the ?DATA CTLDCK card at the beginning of your physical card file.
3. Place the ?ENDCTL card at the end of the deck, unless you plan to use the LDCNTL program again later in the day. Refer to the Creating a Pseudo Card File and Examples of Typical Pseudo Card Decks portions of this section.
4. Place the card file in the card reader and start it.
5. Enter TO APCR to determine if the MCP option APCR has been set.
6. If APCR = 1, the MCP will automatically create and run the pseudo card file. If APCR = 0, you must follow the manual procedure. If you need to use the pseudo card file more than once, you must follow the manual procedure immediately below.

   Manual Procedure:

7. Enter RO APCR to change the value from 1 to 0, if applicable.
8. Enter LD or LDP and transmit. Enter DQ to obtain the deck number; you will need the deck number later.

   NOTE
   With the deck number, you can obtain the status of the pseudo card file by entering the CD, CDP or OL commands. Use CD or CDP for inactive pseudo card files and OL for active pseudo card files.

   a. Enter RN or RNP <deck number> SAVE to activate and process a pseudo card file on the processor you enter this command from.

b. Enter RN or RNP <processor number> <deck number> SAVE to activate a pseudo card file on another processor.

c. Enter RN or RNP = SAVE to activate and process all pseudo card decks on the processor you enter this command from.

d. Enter RN or RNP <processor number> = to activate and process all pseudo card files on another processor.

e. Enter RN or RNP A = to activate and process all pseudo card files on all processors (systems).

NOTE
An active pseudo card file is accessible to all programs on the system on which it was processed.

Automatic Procedure

9. Enter LD or LDP and transmit. Enter DQ to obtain the deck number; you may need the deck number later.

NOTE
With the deck number, use the OL command to obtain the status of the pseudo card files the MCP has activated.

10. Because the MCP automatically activates and processes the pseudo card file, all you need to do is run the appropriate programs to use the pseudo card file.

11. Put the ?ENDCTL card through the card reader, when you are finished with the LDCNTL program.

DELETING PSEUDO CARD FILES

You can use any of the following methods to remove a pseudo card file from the system:

- Use an RD or RDP command to remove inactive pseudo card files that you created manually. Use the DA or DAP command to make the pseudo card file inactive before you attempt to remove it.
- If the MCP option APCR is set, each pseudo card file is automatically removed after it is processed. If the program reading the pseudo card file is discontinued or fails, the pseudo card file remains on the system.
- If a pseudo card file contains a source file for a COMPILE and GO operation, and if the compile fails, or if the system finds an ?END or ?DATA card during flushing of control instructions, the system removes the pseudo card file.
- If the MCP option AURD is set, pseudo card files that contain erroneous combinations of control instructions are automatically removed from the system.

RECOVERY OF PSEUDO CARD FILES

When the system fails in a manner that requires a haltload while pseudo card files are active, recovery of the active pseudo card files is automatic. The system searches the Pseudo-Reader Directory and reinitializes all active pseudo card files. However, it puts the pseudo card files in a not-ready state. You must use the command RN or RNP to reactivate these pseudo card files regardless of the value of APCR.

EXAMPLES OF TYPICAL PSEUDO CARD FILES

Figure 8-1 shows a typical CTLDCK file which contains only one data file and one set of control instructions.

Note the first card - ?DATA CTLDCK. All the cards you want translated into the pseudo card file follow this card.

Note the last card - ?ENDCTL. This indicates that you are finished with the LDCNTL program.

These cards are processed sequentially, in the exact order they are input. If you enter the LD or LDP command at the beginning of the day and plan to use LDCNTL all day, do not use the ?ENDCTL card until you pass the *last card deck* through the reader.

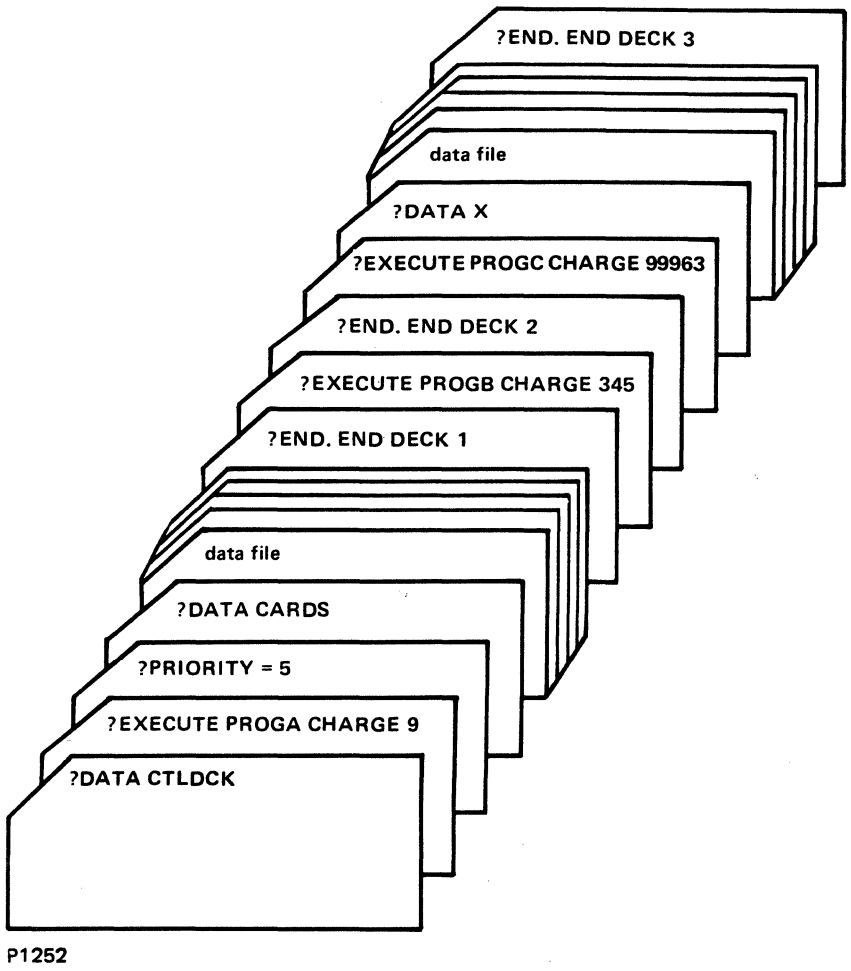


Figure 8-1. Typical Pseudo Card File

You can create more than one pseudo card file out of a single card deck. Each time the MCP reads an ?END card, the pseudo card file is closed. If more cards follow the ?END card, a new file is opened, the MCP assigns the next number in sequence to the new file, and those cards are stored in that file.

A single pseudo card file can contain one or more sets of control instructions or data files. A pseudo card file that contains more than a single set of control instructions or a single data file is called a stacked pseudo

card file. Stacked pseudo card files with more than one set of control instructions will have a value in the link field. The LDCNTL enters this value, which is the record number of the next control card in the file, when it creates a stacked pseudo card file.

Figure 8-2 shows an example of a stacked pseudo card file. Because the MCP processes the pseudo card file serially, the MCP does not process the second EXECUTE request until it reads all cards in the first data file.

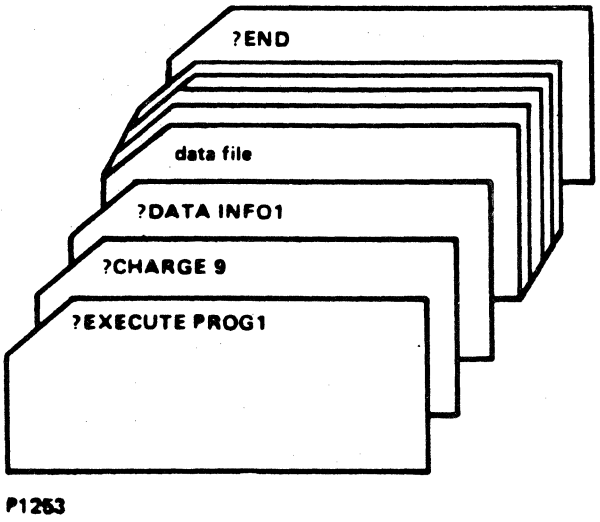


Figure 8-2. Execute Program

FILES AND RECORDS

The pseudo card file is a file consisting of card images, both program data and MCP control instructions, in the following format:

File Attribute	Disk	Diskpack
Record Size	100 UA	120 UA
Blocking Factor	8	6
Records per Area	1000	1200
Number of Areas	100	100

A file must contain an integral number of blocks per area. LDCNTL creates pseudo card files on disk or on disk diskpack in the following format:

Field	Size	Notes
Record Type	1 UN	D for control instructions; 8 for data cards
Flag Digit	1 UN	F (control instructions) specifies link present
Link	6 UN	Zero relative key to next control instruction
Data	96 UA	Card Image
Filler	20 UA	Present for diskpack file only

The record type field contains the first digit of the result descriptor that occurred when the system read the card. D (undigit) means an invalid character appears on the card. The system replaces invalid characters in control instructions with an EBCDIC question mark character (6F).

The flag digit is present in control instructions, when the control instruction link is present. The MCP automatically enters the link field value, when it creates a stacked pseudo card file.

You can recognize control instructions by examining the result descriptor for the record. The first three digits of the result descriptor for control instructions are D40. The system uses an 800 descriptor for all other cards.

FILE ASSIGNMENT

When the MCP processes an active pseudo card file, it processes all control instructions until it reaches a ?DATA statement. A ?DATA statement identifies the name of the data file. When a program needs the data file with that name, the MCP checks both the IOAT and the Pseudo-Reader Directory for the data file name.

Make sure that each data file has a unique name. The MCP assigns files as described in the following paragraphs.

File assignment for a card reader involves connection of the program File Information Block (FIB) of the card file to the appropriate MCP IOAT entry and the initial loading of the program buffers.

Pseudo card file assignment requires the following activities. First, the MCP must obtain a 4 KD main memory buffer for pseudo card file use. In this memory area, the MCP builds an IOAT entry and DFH (Disk File Header) skeleton for the file. Internally, the MCP treats the pseudo card file as a type of disk file. The MCP marks the pseudo card file "in use" in the directory and associates it with the program FIB. Because the external memory area functions as buffers, the MCP modifies the program FIB to reduce the file to a single internal buffer. The MCP reads a block of card images into the external buffer; the MCP fills the program internal buffer, if necessary. This process completes the OPEN request.

READ operations cause the transfer of records from the external buffer to the program buffer, and if necessary, to the program work area. This may be necessary because as the MCP empties the external buffer, it initiates a disk read to refill it. With program execution, the system overlaps all I/O operations.

Reader of File Deallocation

Deallocation of a card reader or pseudo card file means that the system terminates the association between the device and the program using the data file. Deallocation does not mean that the card reader or pseudo card file is deactivated or removed. Deallocation of a pseudo card file occurs when the system detects an EOF (End-of-File) condition, when the program executes a CLOSE request prior to the EOF, or when the program terminates before either event occurs.

EOF detection means that a control instruction (other than a SPO control instruction) appears while the program is associated with the device. Unlike other devices, the system detaches card readers from the program at the point of EOF rather than at a CLOSE request. For pseudo card files, the MCP releases all memory space allocated to the pseudo card file and performs certain maintenance functions on the Pseudo-Reader Directory entry.

If the program executes a CLOSE request or terminates normally before it reaches EOF, the MCP deallocates the card reader or pseudo card file and flushes it to the next control instruction. The system flushes a card reader by passing cards through the card reader until it reaches the next control instruction or until it makes the card reader "not ready."

Pseudo card file flushing achieves similar results, but the system accomplishes it in a different manner. When the pseudo card file is ready for flushing, the MCP examines the next card image in the pseudo card file. If the card is a control instruction, the MCP stops the flushing operation. If that card is not a control instruction, the MCP uses the control instruction link found in the last DATA statement to flush the pseudo card file. This link provides a rapid method of finding the next control instruction without the overhead of reading each record from the disk.

If the link is not present, the MCP attempts to find the next control card by manually reading each card. If the program terminates abnormally (for example, with a DS command), the MCP deactivates the pseudo card file and removes it from the Pseudo-Reader Directory, but does not remove it from the Disk Directory. This makes the pseudo card file "not ready" and you must use the command RN or RNP to reactivate it.

LDCNTL - Punch Backup to Psuedo Card File Conversion

You can use the command CV or CVP to initiate LDCNTL to convert a file in punch backup format to a file in pseudo card file format. This is useful when existing programs generate card files that will be read by subsequent programs though you do not need the cards.

The system always converts punch backup files to pseudo card files on the same media, disk or diskpack. If you specify SAVE, the system keeps the punch backup file in addition to the pseudo card file. If, at the

time of coldstart, you had specified the CHRG ALL, you must now include the CHARGE parameter instruction. The system does not provide special handling of stacker selected card punch records.

Enter CV or CVP <deck number> SAVE or enter CV or CVP <deck number>.

The MCP automatically enters the following parameters into the LDCNTL program:

BASE. + 32	5 UN	CV deck number
	1 UN	flags
	8 bit	CV command
	4 bit	pack media
	2 bit	<unused>
	1 bit	save flag

If you create a user program called LDCNTL, the MCP will also enter these parameters into your program automatically.

SECTION 9
HOST LOAD INTRINSIC PROGRAM (LDHOST)

The Host Load Intrinsic Program (LDHOST) downloads firmware to a programmable controller, a Data Communications Processor (DCP), or a Uniline device. In the case of a B 974 DCP, the LH command invokes the utility program B974LD.

After the load completes, LDHOST displays the release level of the firmware that was just loaded (if you loaded a diskpack/disk channel).

The system does not allow you to download firmware to the primary MCP disk channel.

If the LDHOST program fails to load any device, it leaves the channel of that device inhibited and notifies the system operator that the load failed and that the channel is inhibited. Also, LDHOST displays the segment of device memory that was being loaded when the failure occurred, the result descriptor (R/D), and the number of digits actually transferred.

You do not need to load SMD DLP II. The SMD stands for Storage Module Device, and DLP stands for Data Link Processor.

HOST LOAD FOR B 974 DCPS

The *B 974 DCP Installation and Operation Guide* provides procedures for loading the B 974. In some cases, you will use the command LH to initiate the B974LD utility program. In other cases, you will use the command SO DCP, which does an automatic LH. For more information about the command LH, refer to the *V Series MCP/VS System Software Operations Guide Volume 2: System Commands*. For more information about the command SO DCP, refer to the Data Communications Processor (DCP) section in this manual.

HOST LOAD FOR B 874 DCPS

When you invoke the Host Load program to load a B 874 DCP, the system automatically dumps the S-memory of the DCP. This dump appears as a listing called S-DUMP and a disk file called DCxxpP: where *xx* represents the channel of the B 874 DCP and *p* (lower case) represents the processor number of the host.

AUTOMATIC OPERATION

The MCP can schedule the LDHOST program automatically when a disk or diskpack controller fails. However, the following conditions are necessary before this can happen:

- You must have done a previous LH <channel> <firmware name> to set up an association between the name of the firmware file which is to be loaded to the controller, and the number of the channel to which the controller is connected. Or you must have specified the firmware file name on a DLP record in the system configuration file.
- The firmware file to be loaded to the controller must be on disk.
- The firmware file must be accessible (that is, it must not be on the disk where the controller failed).

If the MCP determines that it can load the controller, the following message will appear on the system console:

CHANNEL <channel number> FIRMWARE LOAD SCHED

In this case, you do not need to do anything. Otherwise, the MCP displays the following message:

CHANNEL <channel number> FIRMWARE LOAD REQD

In this case, you must determine the problem and take the appropriate corrective action.

The MCP automatically enters the parameters that appear in the following table into the LDHOST program.

Table 9-1 LDHOST-MCP Interfaces

	BASE 32 UN=
Firmware ID	6 UA
Primary Channel	2 UN
Alternate Channel	2 UN
B 874 DCP S-Memory Size (HEX)	4 UA
S-Memory Dump Flag	1 UN
Inhibit Channel Flag	1 UN
Hardware Type	2 UN
Unit	1 UN
DCP "WARM" Load Flag	1 UN
Hardware Subtype	1 UN
Filler	1 UN

HOW TO LOAD THE LOS INTO AN INTELLIGENT LASER PRINTER

An Image Page Printer (IPP) is one type of Intelligent Laser Printer. Perform the following steps to load the Loadable Operating System (LOS) into an Image Page Printer (IPP).

1. Make sure that the device is on line and is in the Resident Operating System mode (ROS).
2. To load the LOS, enter LH <channel number>
3. Refer to the *B 9290-30 Intelligent Laser Printing System Operator's Guide* for operating instructions.

SECTION 10
OFF-LINE FIRMWARE LOADER PROGRAM (LOADFW)

You use the LOADFW program before coldstart to load firmware to diskpack controllers, shared system processors (SSPs), and Uniline DLPs. LOADFW performs initialize/verify/relocate (IVR) functions for LAK diskpacks on an individual basis, as well. If you are working from a V 300 processor, you will load this program from a floppy disk. If you are working from a V 500 processor, this program resides on the hard disk of your maintenance processor.

You operate LOADFW from an ODT linked to the console DLP or an OCS linked to a Uniline DLP. First you press the ON LINE switch on the processor to put in console mode, then you load the program from the floppy or the maintenance processor into memory. Once the program is in memory, you execute it using various commands, depending on the functions required.

LOADFW loads the firmware files from tape or a diskpack. If the firmware is on diskpack, the diskpack must have a standard label.

LOADFW loads firmware to the following types of devices:

- Uniline DLPs
- Shared Systems Processors (SSPs)
- Disk or diskpack controllers - Bx383, Bx384, Bx385, B9387

LOADFW can also perform the Initialize/Verify/Relocate (IVR) functions for LAK diskpacks that are shipped in 180-byte format. This is useful for systems that need a diskpack in 100-byte format to use as MCP disk. IVR performs the following functions:

I - Initializes the track and sector structures on the diskpack into a 100-byte format

V - Verifies that the track and sector structures of the diskpack have been initialized properly

R - Relocates and thereby removes bad sectors from use

Note that you can also use DISPKV to IVR diskpacks in LAK mode, which may be more convenient, since it is used while the MCP is running. See the section on DISPKV for more information. This function is not available for MD4 SMD LAK (instead you can use PTDMNO).

Executing LOADFW

Follow these steps to execute LOADFW. The terminal will display various prompts while the program runs.

V 300 — If you are working from a V 300 processor, put the CV321A floppy disk into the primary floppy drive. Make sure the ODT is off-line and in console mode. Do this by pressing the ON LINE/CONSOLE switch on the processor cabinet so that the word CONSOLE on the switch lights up. The processor should be stopped (as indicated by the ODT status heading). If it is still running, press the SPCFY key or enter the TERM command. Issue the following command:

```
LOAD SYS LOADFW; RUN
```

V 500 — If you are working from a V 500 processor, issue the following command:

```
Load Program
    File Name (Def Dir=[Sys] <Program>)__loadfw_____________
    [Load SYS? (Y/N, Def = N)]__Y_____________________
```

Next, press the function key F8 [event]. The system will load the LOADFW program into memory from the floppy disk, or from the maintenance processor, depending on whether you are working from a V 300 or a V 500 processor. This will cause the processor to change from STOPPED to RUNNING, and then LOADFW will begin to execute.

The ODT, or OCS, set up in the SO OCS maintenance panel command will be used for communication. Because of this, the command should be set up beforehand. Be careful that SO OCS is not set to a Uniline OCS that does not have firmware loaded to it.

If you intend to use the ODT for communication, the ODT must be placed on line. You can do this by pressing the ON LINE/CONSOLE switch on the processor cabinet. This will cause the word ON LINE to light up. The program will then begin the dialog by prompting you for the channel and unit of the tape or diskpack that contains the firmware files.

Next, enter the channel and unit number of the tape or diskpack that has the firmware file.

(channel number/unit number)

Next, enter the name of the firmware file to be loaded. LOADFW will find the firmware and load it into memory.

(firmware file name)

After LOADFW notifies you that the firmware has been loaded into memory, enter the channel number of the device to load the firmware to.

(channel number)

After LOADFW notifies you that the firmware has been loaded into the device, you can perform one of the following steps:

- To load the same firmware to another device of the same type, enter another channel number.
- To load a different type of firmware to a different type of device, enter the name of another firmware file.
- To use a different tape or diskpack firmware source, enter the channel/unit.
- When loading a disk controller, you can enter INI or VER and the unit number. INI both initializes and verifies the diskpack; VER verifies it only.
- If you are finished and want to quit the program, enter END.

LOADFW displays error messages when it cannot run successfully. Refer to "Error Messages," in this section, for more information.

Normal Messages

The following messages, which are listed in alphabetic order, report on program status or prompt you to enter information.

Cancelling channel . . .

LOADFW is cancelling the channel. This message is normally sent when SSP DLPs are loaded. There is a 10-second delay to allow the DLP to reinitialize.

<cc/uu> <LAK type> {initialized} with no error(s), no relocate(s)
{verified} with no error(s), no relocate(s)

The LAK has been successfully initialized or verified.

<channel> SSP cleared and unlocked

LOADFW has loaded the SSP successfully. The SSP is now clear and initialized.

Enter channel to be loaded

This is a prompt for you to enter the number of the channel to load.

Enter firmware name

This is a prompt for you to enter the name of the firmware file to be loaded.

Enter input tape or diskpack cc/uu

This is a prompt for you to enter the channel and unit numbers of the tape drive or diskpack that has the firmware files.

Enter: <channel>..... for next channel to load,
<name>..... for new firmware,
<channel/unit>..... for new firmware source,
or END

Or, when you have loaded a disk/pack controller, it prompts:

Enter: <channel>..... for next channel to load,
<name>..... for new firmware,
<channel/unit>..... for new firmware source,
INI <unit>..... to initialize,
VER <unit>..... to verify,
or END

LOADFW displays these two prompts (which appear above) after it completes an operation. It wants to know what you want it to do next. At this point you can instruct it to do one of five things, depending on which of the following you input:

To load the same firmware to another channel, enter the channel number:

<channel number>

To load another firmware file, enter the name firmware file name:

<firmware file name>

To enter a new channel and unit number of the tape or diskpack that has the firmware file:

<channel number/unit number>

To initialize an LAK, enter the INI command and its unit number:

INI <unit>

To verify an LAK, enter the VER command and its unit number:

VER <unit>

To quit the program, enter END:

END

{ Initialization}
{ Verification } <percent> complete
Time remaining: <minutes> minutes, <seconds> seconds

During a LAK initialization or verification, you receive this progress report.

Reading firmware ...

LOADFW is searching the input medium for the requested firmware file.

Loading firmware ...

LOADFW is loading the requested firmware.

(<release code>) phase {1 } {2 } {3 } loaded {all}

LOADFW has completed the requested firmware load operation.

EOJ LOADFW

LOADFW has gone to End-of-Job.

Error Messages

LOADFW displays the following error messages.

<cc/uu> invalid label
<cc/uu> not ready
<cc/uu> unknown input medium tape
<cc/uu> does not have <firmware file name>
<cc/uu> read error continue?
Corrupted directory/header link

LOADFW will display one the these messages when it cannot load firmware. The actual message displayed indicates why it cannot perform the function.

<cc/uu> not LOADMP, PACKUP or SYSTEM/COPY tape

The device on channel/unit has the wrong tape format. The tape must have a LOADMP, PACKUP, or SYSTEM/COPY tape format. The SYSTEM/COPY format is recommended.

Enter input tape or diskpack cc/uu again

The program displays this message when it wants you to reenter the channel and unit number. Frequently, this message is displayed after another error message that refers to a previously-used tape or diskpack.

<cc/uu> got invalid I/O on load

An I/O error occurred when LOADFW attempted to load the firmware.

<cc/uu> is not a LAK or initializable media

Either the device on that channel is not LAK diskpack in 100-byte format, or it is the wrong type of device.

LOADFW cannot initialize/verify MD4 SMD disks. To initialize MD4 disks, use the Peripheral Test Driver (PTD) program SMDIVR.

<cc/uu> <LAK type> {initialized}
{verified} with <number> errors, <number> relocates

After the program finishes an initialize or verify operation, it displays this summary.

DLP not an HT, SSP, or Uniline - enter channel:

LOADFW determines the device type with a TEST ID operation. The program can load either a Host Transfer, SSP or Uniline DLP. This message usually indicates that the requested channel is other than one of these three types. The message may be displayed because the DLP is hung. Refer to Section 4 for information on clearing a hung DLP.

Error on sector <number> in cylinder <number> of <max cylinder number>
Enter (I)gnore, (R)elocate, (E)nd or blank for retry

During an initialize or verify, LOADFW found an error on the hardware. You must now enter:

I — Ignore and continue with the next cylinder.

R — Relocate the bad sector.

E — End to terminate the operation.

Blank to retry the operation.

Fatal IOP error occurred
<cc/uu> IOP R/D = <R/D>

LOADFW encountered an IOP error. The result descriptor (R/D) involved appears on the second line of the message.

Segment <number> not loaded - <number> of <number> digits passed

LOADFW stopped just before the indicated segment number. The number of digits passed indicates the amount of firmware loaded.

SECTION 11
FLOPPY DISK COPY UTILITY PROGRAM (MDCOPV)

The MDCOPV utility copies files to and from floppy disks and performs floppy disk maintenance functions.

The information is stored on the floppy disks using the Industry Compatible Minidisk (ICMD) format or Micro-Minidisk Unit (MMDU) format, depending on whether you have a processor that uses 5.25- or 8-inch floppy disks. MDCOPV automatically determines the type of system it is running on and selects the corresponding storage format, ICMD or MMDU. MDCOPV performs three basic functions:

- Copy Functions - copies files or an image of an entire floppy disk to and from a floppy disk.
- Analysis Functions - initializes floppy disks and removes files. Reports the amount of space used by files and the amount available.
- Conversion Functions - converts object code or A Series S-code files to MMDU compatible format.

Supported Hardware

MDCOPV functions under the release level of the MCP/VS operating system covered in this document. The only V series systems that require MDCOPV are V 300 series systems. MDCOPV can also function in other environments, subject to the following system and hardware requirements:

Processor	DLP	Floppy Disk Format	Storage Capacity	Required Unit Number
V300/B4900	UC-DLP	5 1/4 MMDU	(640KB)	2 or 3
B2900/3900	UC-DLP	5 1/4 ICMD-LAK	(246KB)	2 or 3
B2900/3900	UC-DLP	8 ICMD	(246KB)	2 or 3
pre-900 sys	ICMD-DLP	8 ICMD	(246KB)	0 or 1

Due to hardware limitations, MDCOPV can access only floppy disk drive units 0 and 1 when running through a control or ICMD-DLP, or units 2 and 3 when running through a UC-DLP.

All copy functions require an initialized floppy disk. Refer to the IN function in the Analyze Floppy Disk command syntax later in this section. The ICMD-LAK format floppy disks cannot be initialized by the MDCOPV utility. The 'DIAG20' maintenance program, which runs in the universal console, must be used to initialize these floppy disks.

Two records must be included in the MCP configuration file for the MCP to access the floppy disk drives. One is the DLP CONSOLE record; the other is the UNIT NST record. The input and output channel numbers will be that of the console DLP for MMDU format floppy disks.

Valid unit numbers for floppy disks attached through controls or HT-DLP are '0' and '1'. Valid unit numbers for floppy disks attached through a UC-DLP are '2' or '3'.

Format Information

Industry Compatible Minidisk (ICMD) format floppy disks must use the industry compatible format for floppy disks described in IBM publication GA21-9190-1. These floppy disks have a maximum record size of 128 bytes and a blocking factor of 1.

Industry Compatible Look-a-Like (ICMD-LAK) format floppy disks are a combination of MMDU format and ICMD format. The floppy disk is initialized in MMDU format (256 bytes per sector) and used as if it were ICMD format (128 bytes per sector). Micro-Minidisk Unit (MMDU) format floppy disks have a maximum record size of 256 bytes and a blocking factor of 1.

Operating Instructions

To use MDCOPV, you must first execute the program with the MCP command EX, and then input commands to it using the MCP command AX. The AX command associates commands that are being input to the executed program with a mix number. So, when you execute MDCOPV, determine its mix number. Use the following syntax to execute MDCOPV:

```
EX MDCOPV
```

After it is executed, MDCOPV displays the following message to indicate that it is ready for you to AX the commands.

```
ENTER INPUT PARAMETERS FOR MDCOPV
```

After the above message is displayed, get the mix number, and input the commands you want in the following manner:

```
<MIX>AX"CH2 12 MINI/FINBAR HOMER BYPASS"
```

Once you enter the commands and the processing completes, MDCOPV displays the same message again, indicating that it is ready for more input.

```
ENTER INPUT PARAMETERS FOR MDCOPV
```

At this point, you can input another command string, or you can input this command to stop processing:

```
<MIX>AX END
```

While processing, MDCOPV may display prompts and messages. These are documented later in this section.

Executing MDCOPV and Setting its Switch Settings

Program switch 1 (SW1) sets the length of the header record length field, which is used by the Purge and Remove functions.

SW1 0 sets the header record length to 80 bytes. This is the default. Simply execute MDCOPV without setting any switches if you want this record length.

SW1 1 sets header record length to 128 bytes for the ICMD format, or 256 bytes for the MMDU format.

Here is an example of how to execute MDCOPV and set its SW1 switch setting so that the header record length is 128 for ICMD or 256 bytes for MMDU.

```
EX MDCOPV; VA 1 100000
```

Copy Functions

The following paragraphs discuss the various copying functions performed by MDCOPV. They address copying files to and from a floppy disk and copying pseudo card reader files. Refer to the section below for information about copying floppy disk image files.

In all cases, the MDCOPV copy function requires that the floppy disk be initialized before it can be used. Refer to the IN parameter in the Analyze Floppy disk command syntax below for more information about initializing floppy disks.

Copying Files *to* a Floppy Disk

When you copy files to the floppy disk, MDCOPV checks for the largest available area on the floppy disk. If no space is available, a message appears on the ODT. Otherwise MDCOPV writes the file. If the file is too big, MDCOPV writes what it can, and then requests that you insert another floppy disk so that it can continue writing the file.

For ICMD and ICMD-LAK format floppy disks, the maximum record size allowed is 128 bytes. MMDU format allows a maximum record size of 256 bytes. Care must be taken in copying files to the floppy disk with record sizes greater than these maximums. MDCOPV displays a warning message stating that the record size is greater than the maximum and that, if allowed to continue, record truncation will occur. To continue at this point, you must AX a "Y" to continue or an "N" to stop processing.

Copying Files *from* a Floppy Disk

When copying files from the floppy disk to disk or to diskpack, the copied files are created according to the parameters input when you execute MDCOPV. If you do not use output file specifications, the new file is created using a default record size, blocking factor, number of areas, and records-per-area.

MDCOPV can copy files into pseudo card reader (PCR) format. This lets you create data files or execution decks on floppy disk. Then, when these files are copied from the floppy disk, the MCP can activate them through standard pseudo card reader processing. MDCOPV will interrogate the file being copied for control records. If the first record is not a control record (?DATA or ?EX for example), MDCOPV puts a ?DATA <file name> record at the front of the pseudo card file. The <file name> is that specified in the input string. MDCOPV also creates a ?END record at the end of the file if one is not there already.

Duplicating Floppy Disks

This function makes exact copies of a floppy disk. MDCOPV starts at the first sector on the floppy disk and copies all the sectors thereafter. The label, tables, and data are identical after the copy is completed.

Copy Function Command Syntax

Use the following command syntax to copy files to and from the floppy disk. The copy function syntax can copy files from 100-byte disk and 180-byte diskpack to the floppy disk. Conversely, it can copy files from the floppy disk to 100-byte disk and 180-byte diskpack media.

This syntax can also be used to copy pseudo card reader files and floppy disk image files to and from the floppy disk.

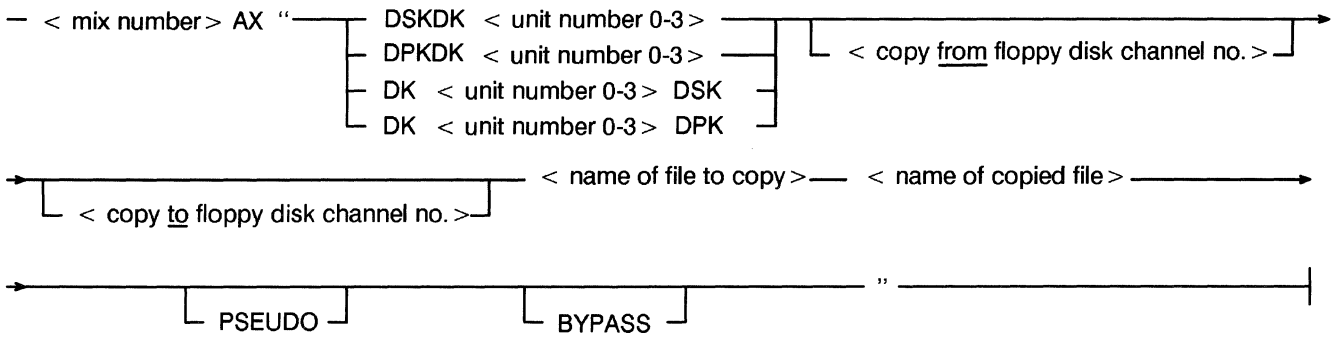


Figure 11-1. MDCOPV Floppy Disk Copy Syntax

<mix number> AX "

After you execute MDCOPV, get its mix number, since the commands are input to the program using the MCP command AX. Note the use of the quotation marks.

DSKDK<unit number 0-3>

This indicates that the file is to be copied from 100-byte disk to the floppy disk drive with the specified unit number.

DPKDK<unit number 0-3>

This indicates that the file is to be copied from 180-byte diskpack to the floppy disk drive with the specified unit number.

DK<unit number 0-3>DSK

This indicates that the file is to be copied from the floppy disk drive with the specified unit number to 100-byte disk.

DK<unit number 0-3>DPK

This indicates that the file is to be copied from the floppy disk drive with the specified unit number to 180-byte diskpack. You can copy the file to a diskpack family by using a multifile name.

<copy *from* floppy disk channel number>

This indicates the number of the channel to which the floppy disk drive unit is connected. This channel must be declared to the system with the hardware type NST (non-status device). The input and output channel numbers will be that of the console DLP for MMDU format floppy disks.

<copy *to* floppy disk channel number>

This indicates the channel number to which the floppy disk unit is connected. The floppy disk channel must be declared on the system and have a hardware type of "NST" (non-status device).

The default output channel number is the input channel number specified.

<name of file to copy>

The file name is in the form <multifile name>/<file name>, where the <multifile name>, which can be the name of a diskpack family, is optional. The <file name> can be from 1 to 8 characters in length for a file on the floppy disk. It can be from 1 to 6 characters in length for a file on disk or diskpack. A <multifile name> entry may be specified for disk diskpack or floppy disk files.

If the <multifile name> is used, it may be any name from one to six characters in length. When using a multifile name, the entire command string must be enclosed in quotation marks. The <multifile name> for floppy disks is the volume ID referenced in the volume label.

The following default values are used if the multifile name is not use:

floppy disk - all blanks (EBCDIC)

DPK - system default diskpack

<name of copied file>

The name of the copied file takes the form <multifile name>/<file name>, where the <multifile name> is optional.

See the preceding <name of file to copy> regarding multifile names.

PSEUDO

This is used only when copying files to disk or diskpack. It indicates that the output file being copied is in pseudo card reader deck format. This assigns the file a pseudo deck number (#number), thus enabling the MCP to activate it in normal pseudo card reader processing.

BYPASS

This applies only to files on floppy disk. After the bypass indicator has been set, it cannot be reset. The bypass option provides two capabilities.

- It enables the user to set the bypass indicator in the File Header when copying a file to the floppy disk.
- It also allows the user access to a floppy disk file when the bypass indicator is set.

"

If a quotation mark begins the command string, it must end with one, too. ∎

Duplicating Floppy Disks with the Copy Function

Use the following command syntax to duplicate floppy disks.

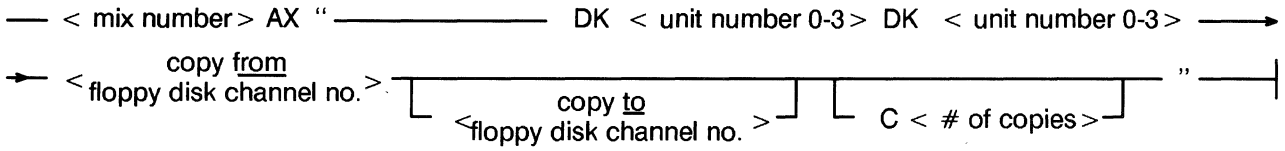


Figure 11-1a. MDCOPV Floppy Disk Duplication Syntax

DK<unit number 0-3>DK<unit number 0-3>

This indicates that the entire floppy disk is to be copied from the floppy disk drive with the specified unit number to another floppy disk drive with the specified unit number.

- C<# of copies>

  This field indicates the number of times a floppy disk is to be duplicated. Copies are specified by entering a "C" followed by the number of copies you want. For example, C5 would create five copies. The value can be up to three digits in length. One copy is the default.

File and Floppy Disk Analyze Functions

Use the following command syntax to analyze the status of the floppy disk. This command syntax does such things as display and rename the volume and file names, removes files and initializes the floppy disk. This list outlines the functions performed by this syntax:

- Lists the files on the floppy disk.
- Displays the volume ID of the floppy disk.
- Prints a status listing of the floppy disk.
- Changes a file name on the floppy disk.
- Renames the volume ID of the floppy disk.
- Purges the floppy disk.
- Removes a file from the floppy disk.
- Initializes a floppy disk.

You must first execute MDCOPV before you input these commands. After you execute the program, get its mix number since you need it to AX these commands to the program.

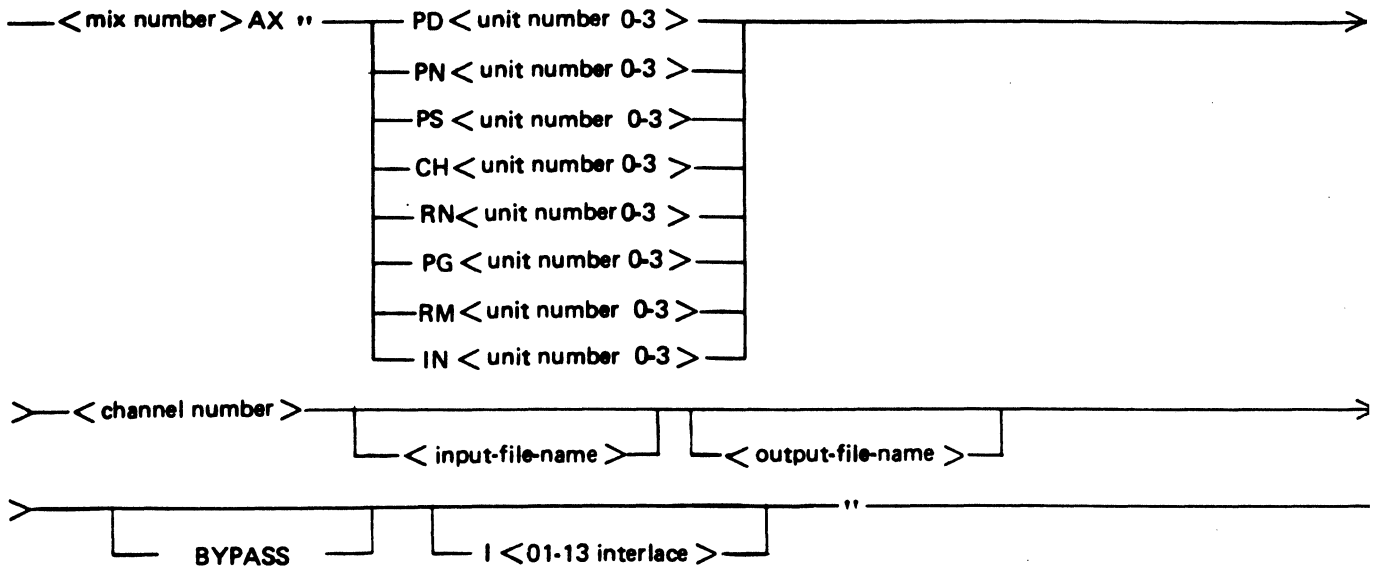


Figure 11-2. MDCOPV Flexible Disk Analyze Syntax

<mix number> AX "

After you execute MDCOPV, get its mix number, since the commands are input to the program using the MCP command AX. The commands you input to MDCOPV must be enclosed in quotation marks.

PD<unit#>

This displays what files are on a floppy disk.

The names of all the files on the floppy disk are displayed on the system ODT. The floppy disk volume ID is also displayed. If the floppy disk does not contain any files, a message indicating that is displayed.

PN<unit#>

This displays the volume ID of the floppy disk.

The volume ID of a floppy disk (multifile name) is displayed on the system ODT. This name is required when any file maintenance or copy functions are being performed. The default volume ID is all blanks.

PS<unit#>

This prints a status listing of the floppy disk. This listing includes the volume ID, the file names, the beginning and ending of each file, and the total space available.

A printer backup file with the name MDSTAT is created giving the exact status of a floppy disk. This listing contains a detailed analysis of the floppy disk directory along with pertinent volume label information. It contains all the information about the files on the floppy disk: for example,

BOE (beginning-of-extent), EOE (end-of extent), bypass indicator, and EOD (end-of-data). Also, the total available space and largest available area on the floppy disk is printed.

CH<unit#>

This changes the name of the file on the floppy disk.

RN<unit#>

This renames the volume ID.

Depending on the manufacturer, preinitialized floppy disks have a volume ID of either all blanks or 'IBMIRD'. These names can be changed.

PG<unit#>

This purges all the files on the floppy disk, and then returns all the label sectors to their original preinitialized state. This only accesses the floppy disk index track sectors 7 through 25; that is, only the floppy disk directory is changed, all the other sectors remain the same.

The header record length field in all file headers will be set to 80 bytes unless you have set switch 1 when you executed MDCOPV. If so, the length will be set to 128 bytes for ICMD format floppy disks or 256 bytes for MMDU format floppy disks.

RM<unit#>

This removes files on an individual basis. Multiple or group removes are not allowed. When removing a file from the floppy disk, the file's directory header along with all the data sectors are returned. MDCOPV then tries to combine all contiguous available sectors into one large available area. If no sectors adjacent to the returned area are available, the area is marked available and possible checkerboarding can occur.

This function has the capability of removing a file which contains invalid extents. MDCOPV does not try to combine this space with any other available areas. The file header is written with the extents equal to their preinitialized values. Removing a file in this manner can cause a missing disk situation.

Note that the header record length field in the removed file header will be set to 80 bytes unless you have set switch 1 when you executed the program. If you set this switch, the length will be set to 128 bytes for ICMD format or 256 bytes for MMDU format.

IN<unit#>

This formats a floppy disk; floppy disk formatting (initializing) destroys all data on the floppy disk. This function allows unreadable floppy disks to be initialized and made readable.

ICMD-LAK format floppy disks cannot be initialized by MDCOPV. The 'DIAG20' program, released on the maintenance test program tape, must be used.

<channel number>

This indicates the channel number of the floppy disk drive you are using.

<unit 0-3>

This indicates the unit number of the device. Note that due to hardware limitations, the unit numbers must be 0 or 1 when using a HT-DLP or 2 or 3 when using a UC-DLP.

<input-file-name>

This is required only for the CH, RM, and RN functions. It indicates the name of the file to change (CH) or remove (RM). Or, it indicate the volume ID to rename (RN).

<output-file-name>

This is required only for the command that changes the name of a file (CH). It indicates the file's new name. This entry may be any name from one to eight characters in length. A multifile name entry is ignored when using the CH function.

BYPASS

This applies only to the commands that remove (RN) a file, or change its name (CH).

This lets you remove or change a file name that has its bypass indicator set. If the file being accessed has the bypass indicator set, and this keyword is not used, a "file not on floppy disk" condition will occur.

NOTE
The bypass indicator cannot be set or reset by the CH function.

I<01-13 interlace>

This applies only to the "IN" function. It specifies the data pattern to be written during the floppy disk initialization. You can specify the interlace code by entering an "I" followed by the 2 digit interlace number, which can range from 01 through 13.

The default interlace code for a floppy disk connected through a DLP (the MMDU format) is I13. The default for a floppy disk connected through a standard I/O control (the ICMD format) is I01.

"

A quote must end the command string input to MDCOPV.

Copy Function For Floppy Disk Image Files

An image file is a single file on 100-byte disk that contains all the information and files for a floppy disk. In some cases, image files are included on release tapes for you to copy to floppy disk.

The following input string must be input to MDCOPV to perform a copy of the whole floppy disk image file to or from the floppy disk.

In the following syntax, the <number-of-flexible-disk copies> parameter is used to specify where the image file is going. A value of 00 writes the file to 100-byte disk. Any other value causes MDCOPV to read an image file on disk and copy it to floppy disk and to repeat this process the specified number of times.

This feature is for use by Unisys Maintenance personnel.

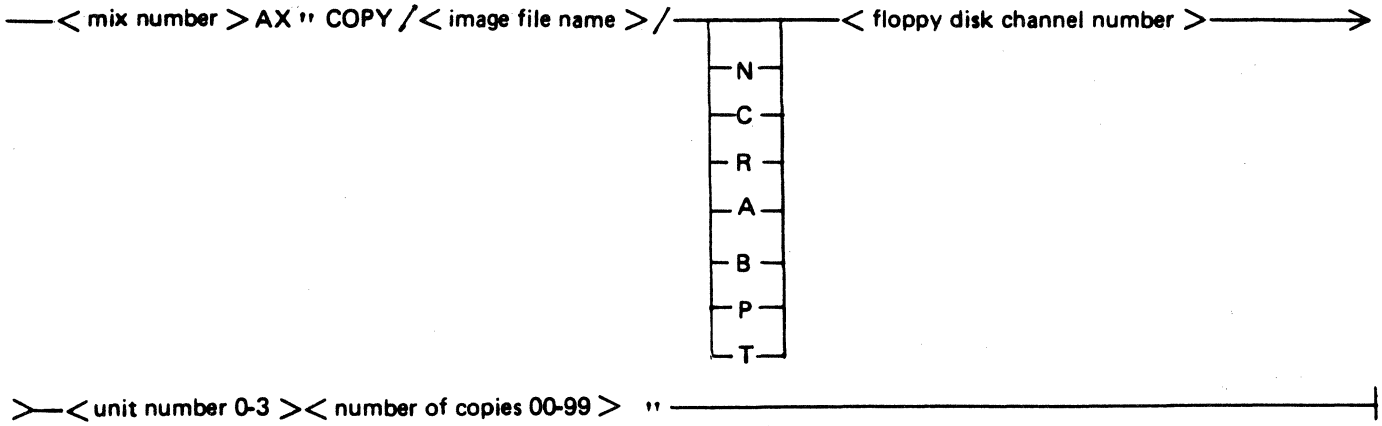


Figure 11-3. MDCOPV Floppy Disk Image File Syntax

<mix number> AX "

After you execute MDCOPV, get its mix number, since the commands are input to the program using the MCP command AX. Note the use of the quotation marks.

COPY/<image-file-name>

This indicates the name of the file to copy. It can be 1 to six characters long.

N

This indicates not to check the part number.

C

This initiates a compare pass after writing each copy

R

This recycles used floppy disks for new copies

A

This sets the options N, C and R

B

This sets the options C and R

P

This protects the floppy disk master file or copies (inserts 'P' in headers, includes option C).

T

This terminates without removing the floppy disk (one copy, includes options N, C and R)

<flexible-disk-channel>

This indicate the channel number of the floppy disk drive.

<unit 0-3>

This indicates the unit number of the floppy disk drive.

<number of copies 00-99>

This indicates the number of copies you want. Use a two digit number to indicate the number of copies.

A figure of 00 indicates that the file is to be copied from the floppy to 100-byte disk.

"

A quote must end the command string input to MDCOPV.

Conversion Functions

This function converts 100 character per record code files or 128 character per record S-code files into a 256 character per record disk or diskpack file. The conversion function can also copy this file to a 5 1/4 floppy disk, or the copy function can be used.

This feature can be used to convert the following types of files:

1. It can convert an object code file created by the BPL compiler with the 'MCPP' compiler option set.
2. Object code file created by the ASMBLR compiler.
3. Any 128 character record S-code file create by the PTL compiler on an A Series System. The following syntax invokes the floppy disk file conversion function.

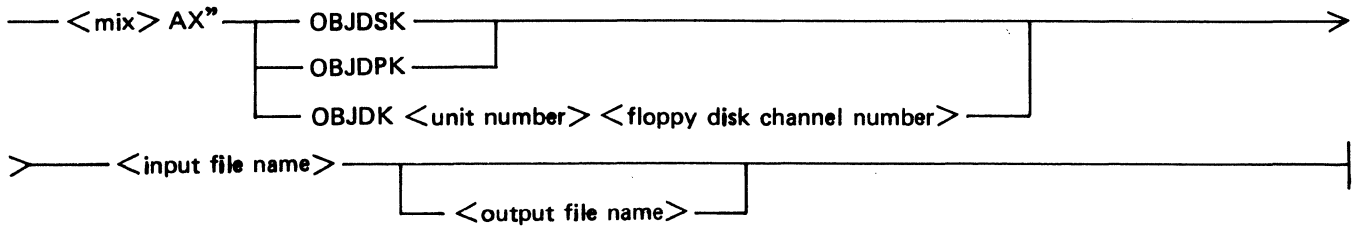


Figure 11-4. MDCOPV File Conversion Syntax

<routine type>

This is a 6 character concatenation of the literal "OBJ" and a 3 character output device mnemonic. The following values are the only ones supported:

OBJDK<unit number> — The object or S-code file will be converted to 256 character record format and then copied to the flexible disk on the specified unit number. Note that the channel number of the floppy disk drive must be specified for this option. See below.

OBJDSK — The object or S-code file will be converted to a 256 character per record format on disk.

OBJDPK — The object or S-code file will be converted to a 256 character per record format on diskpack.

<floppy disk channel number >

This specifies the channel number of the floppy disk drive. It is required if the routine type OBJDK is used. It is invalid in all other cases.

<input file name>

This identifies the input object or S-code file. A multi-file ID may be specified if the file is on diskpack.

<output file name>

This specifies the name of the output file. A multi-file ID can be used for diskpack and floppy disk files. This entry is required for output to floppy disk but is optional for disk or diskpack output files. The default output file name is the same as the input file name, except that the last character of the name is a z.

Examples

Here are a number of examples of how to input commands to MDCOPV.

Copying files to a floppy disk.

```
31 AX "DSKDK2 24 HSTLDP HSTLDP "
31 AX "DSKDK3 13 HSTLDQ SYSTEM/HSTLDQ BYPASS "
31 AX "DPKDK3 13 HERBIE/HSTLDN MINI/HSTLDN BYPASS "
```

Copying files from a floppy disk to disk and diskpack.

```
31 AX "DK2DSK 24 HSTLDP HSTLDP "
31 AX "DK3DSK 13 SYSTEM/HSTLDQ HSTLDQ BYPASS "
31 AX "DK3DPK 13 MINI/HSTLDN HERBIE/HSTLDN PSEUDO BYPASS "
```

Copying an entire floppy disk to another floppy disk.

```
31 AX "DK2DK3 7 7 "
31 AX "DK2DK3 7 C99 "
31 AX "DK3DK2 7 7 C32 "
```

Copying image files to and from floppy disk.

```
31 AX "COPY/PANDLP/A13300 "
31 AX "COPY/PANEL1/C13201 "
```

Analyze Functions

```
31 AX "PD3 13 "
31 AX "PN3 13 "
31 AX "PS2 13 "
31 AX "CH2 12 MINI/FINBAR HOMER BYPASS "
31 AX "RN2 12 MYMINI "
31 AX "PG3 12 "
31 AX "RM2 13 MINI/HOMER BYPASS "
31 AX "IN3 15"
```

Here are two examples of how to invoke the conversion function:

```
31 AX "OBJDK3 10 M68FVR PTDISK/M68FVR"
31 AX "OBJDSK LOAD0B LOADFW"
```

File Equate Information

Table 11-1 lists the internal and external file names used by MDCOPV:

Table 11-1 File Names for MDCOPV

Internal Name	External Name	Comment
MDSTAT	MDSTAT	Status Report Listing
PAKFIL	PAKFIL	Output Disk/Pack file
PSRFIL	#<file number>	Pseudo card reader file
INPAK	INPAK	Input Disk/Pack file
FLCFIL	FLCFIL	Image file - ICMD format
COPFIL	COPFIL	Image file - MMDU format
Object Code input	OBJINP	OBJINP
S-code input	SCODEI	SCODEI
Converted Object Output	OBJOUT	OBJOUT

MDCOPV Messages

The messages displayed by MDCOPV fall into these three categories:

- Informational messages - These messages indicate the completion of a function and require no operator intervention.
- Operator Prompts - These messages prompt you for some type of input.
- Error messages - These are error messages that do not require operator intervention. After displaying one of these messages, the program will usually abort the current operation and return to the 'waiting input parameters' state.

Informational Messages

The following messages are strictly informative. No response is required. Each message is followed by a description.

<file name> CHANGED TO <file name>

The specified <file name> was successfully changed to a different <file name>. Output by the 'CH' function.

<volume-ID> CHANGED TO <volume-ID>

The specified floppy disk <volume-ID> was successfully changed to a different <volume-ID>. Output by the 'RN' function.

<file name> COPIED TO MINIDISK AS <volume ID>/<file name>

A disk or diskpack file has been successfully copied to the floppy disk identified by <volume-ID>/<file name>.

<volume-ID>/<file name> COPIED TO <media> AS <file name>

This message is displayed after a floppy disk file identified by <volume-ID>/<file name> has been successfully copied to either disk (<media> DSK) or disk diskpack (<media> DPK).

MINIDISK COPY COMPLETED ON <channel number/unit number>

The duplicate floppy disk function has completed normally.

MINIDISK COPY #<copy number> COMPLETED ON <channel number/unit number>

The duplicate floppy disk function has completed copy number <copy number>. The floppy disk on the specified <channel number/unit number> should be removed and replaced by a scratch floppy disk if more copies were requested.

<channel number/unit number> MINIDISK PURGED

The floppy disk on the specified channel and unit has been successfully purged.

<file name> REMOVED FROM MINIDISK

The floppy disk file <file name> has been successfully removed from the floppy disk.

<channel number/unit number> VOLUME LABEL ID<volume-ID>

This message is output by the 'PN' function and shows the <volume-ID> of the floppy disk.

<interlace type> MINIDISK INITIALIZED ON <channel number/unit number>
<bad tracks message>

These messages are displayed after a floppy disk has been successfully initialized. The <interlace type> will be either 'INTERLACED' or 'SEQUENTIAL', depending on the <interlace code> used during initialization. The <bad tracks message> will be one or more of the following:

NO BAD TRACKS
FIRST BAD TRACK =nn
SECOND BAD TRACK = nn

Input Prompts

When MDCOPV displays the following prompts, you must AX a response before it continues processing.

ADDITIONAL <memory required> KD REQUIRED FOR READ BUFFER
REEXECUTE MDCOPV WITH ADDITIONAL MEMORY

Because of the size of the input/output file buffer, additional memory is required for processing this function. Reexecute MDCOPV with the following syntax: EX MDCOPV MEMORY + <memory required>.

CONTINUATION MINIDISK <sequence number> REQUIRED
LOAD AND <MX>AXDK<UNIT#>

A floppy disk file that extends across more than one floppy disk is being read or written, and the next floppy disk in sequence is required. If creating a file, the <sequence number> is the number that will be written to the floppy disk identified by 'DK<UNIT#>'. If reading the file, the <sequence number> identifies the next floppy disk required. Load the appropriate floppy disk and input the appropriate ODT response.

ENTER INPUT PARAMETERS FOR MDCOPV

MDCOPV displays this message once it is executed. Once you have entered the input parameter string, and the specified function is completed, MDCOPV will again display this message and wait for input. This will continue until you enter '<MIX>AX END' to end processing.

INVALID R/D RETURNED ON SECTOR WRITE
DIR R/D: <result descriptor> DIR EXT R/D: <extended descriptor>
ENTER R RETRY OR N NO RETRY

An attempt was made to do a write "deleted" on a faulty sector. The result descriptors being displayed were received after 10 retries. At this point enter 'N'. MDCOPV will stop the current function and return to the 'waiting input parameters' state.

MINIDISK COPY #<copy number> COMPLETED ON <channel number/unit number>
MOUNT A SCRATCH MINIDISK ON <channel number/unit number>
AND ENTER GO WHEN READY

The duplicate floppy disk function has completed copy number <copy number>. The floppy disk on the specified <channel number/unit number> should be removed and replaced by a scratch floppy disk and <MIX>AXGO should be entered.

READ EXTENDED FAILED R/D
DIR R/D: <result descriptor> DIR EXT R/D: <extended descriptor>
ENTER R = RETRY OR N = NO RETRY

The <result descriptor> and <extended descriptor> displayed were received when attempting to perform a Read Extended Status. An invalid result descriptor was received on the previous I/O, causing MDCOPV to attempt a Read Extended Status to determine the exact problem. If 'N' is entered, MDCOPV will abort the current function and return to the 'waiting input parameters' state.

RECYCLE USED MINIDISK OK?
ENTER Y(ES) OR N(O)

The floppy disk to be used for making a copy of an image file contains valid data. If it is all right to overwrite this floppy disk then enter <MIX> AX Y otherwise enter <MIX> AX N.

INPUT REC SIZE EXCEEDS <record-size> BYTES
TRUNCATION WILL OCCUR IF COPIED TO MINIDISK
CONTINUE Y = YES N = NO

WARNING

This message indicates that the disk or diskpack file that is being copied to the floppy disk has a record size greater than 128 for ICMD format or 256 for MMDU format. If the copy is allowed to continue, the records written to the floppy disk will be truncated and data will be lost.

INVALID RD RETURNED ON <channel number/unit number><result descriptor>
ENTER RRETRY, OR NNO RETRY

An I/O error occurred while reading or writing to the floppy disk on <channel number/unit number>. If you enter 'N', MDCOPV will display the following messages and wait for input:

COPY TO MINIDISK ON <channel number/unit number> FAILED
MOUNT A SCRATCH MINIDISK ON <channel number/unit number>
AND ENTER GO WHEN READY

Error Messages

After displaying one or more of the following error messages MDCOPV will abort its processing and return to the 'waiting input parameters' state.

ABORTED DUE TO BAD SECTOR

This message is displayed if you enter a 'NO' response to the message 'ENTER R = RETRY OR N = NO RETRY'. A memory dump also takes place.

<file name> ALREADY PRESENT ON MINIDISK

The disk or diskpack file to be copied to the floppy disk already exists. Either copy the file to another floppy disk, or remove the file from the floppy disk by using the 'RM' function.

BAD FILE - UNDIGITS IN START ADDRESS

The object code file input to the conversion function has undigits in the start address at BASE.+94. This file cannot be used.

BAD FILE —No Start Address at 94

The object code file input to the conversion function has zero in the start address at BASE.+94 or the address controller at the start address is not zero. This file cannot be used.

DUPLICATE MEDIA NOT ALLOWED

The 6 characters that indicate the direction of the copy cannot contain duplicate hardware types.

FILE-ID NOT CHANGED

The 'change to' <file name> specified for this 'CH' function already exists on the floppy disk.

INPUT FILE NOT ON MINIDISK

The input file specified cannot be found on the floppy disk. The <file name> and bypass indicator must match before access is allowed.

INV CONTINUATION FILE SEQ#, IS nn, EXPECTED nn

A multivolume floppy disk file was encountered and the wrong continuation floppy disk was input to MDCOPV. The displayed sequence numbers are the one that MDCOPV expected and the one actually received.

INV HEADER EXTENTS, SECTOR <sector number>

The File Header <sector number> displayed has overlapping or invalid extent fields. This floppy disk is unusable by MDCOPV until a purge or initialize is performed on the floppy disk.

INV INPUT BLOCK SIZE > 99

The records-per-block value in the disk file header of the input disk or disk diskpack file is greater than 99. If you want to copy this file to the floppy disk, it must be reblocked.

INV MULTI VOL SEQ# <header-volume-seq-number>

The header volume sequence number found in the input File Header label is invalid. MDCOPV expects a blank volume sequence number and blank multi-volume indicator for single floppy disk files, or a 'C' or 'L' in the multi-volume indicator and '01' through '99' in the volume sequence number for multi-volume floppy disk files.

INV OUTPUT NUMBER OF COPIES

The number of copies parameter was expected and not found, or the parameter is invalid in some way. The correct syntax for this parameter is 'C' followed by a 1 to 3 digit number. There is no space between the 'C' and the number, e.g. 'C10' if 10 copies are required.

INVALID CONTINUATION MINIDISK REC SIZE

The record size specified in the File Header of a continuation floppy disk does not equal the original record size.

INVALID IMAGE-COPY ROUTINE INPUT ENTERED

Either the name of the image file is missing or the second '/' delimiter was not found.

INVALID INPUT ENTERED

The string you entered in response to the prompt 'ENTER INPUT PARAMETERS' contains an invalid token, for example, a multifile name which is greater than 6 characters.

INVALID INPUT FILE RECORD SIZE

The code file input to the conversion function does not have a record size of 100 characters (object code) or 128 characters (S-code).

INVALID INPUT FUNCTION

The input hardware type you specified is invalid, or the 3 character analyze function is not recognized.

INVALID INPUT ROUTINE TYPE

The first item in the input parameter string is not 3 characters (for the analyze function), 6 characters (for the copy function), or 'COPY/' for the image-file copy function.

INVALID INTERLACE CODE

The <interlace code> entered as part of the 'IN' function parameter string was invalid. The correct format is 'Inn' where 'nn' can range from 01 to 13.

INVALID MINIDISK CC/U

The channel/unit combination entered is not correct.

INVALID MINIDISK CHANNEL #

The channel number entered is not present or is invalid.

INVALID MINIDISK UNIT #

The unit number entered is not in the range of 0 through 3.

INVALID OUTPUT FILE-ID

An <output-file name> was expected but was not found.

INVALID OUTPUT FUNCTION

The output hardware type specified is invalid.

INVALID OUTPUT REC SIZE

The record size value in the label for this floppy disk file is not mod 4. The copy function is aborted.

INVALID '(O)CCUNN' INPUT

The parameters for the Image file copy function were missing or invalid.

INVALID S-CODE FILE FORMAT

The input S-code file has a record size of 128 characters but does not contain "PTL" in the first record and is therefore not a correctly formated S-code file.

I/O ERROR IN TRACK ZERO
BAD MINIDISK - INITIALIZE ABORTED ON <channel number/unit number>

MDCOPV was unable to initialize the floppy disk because of an error in track zero. The floppy disk is unusable and should be disposed of.

MINIDISK MEDIA NOT PRESENT

A floppy disk hardware specification (DK<unit number>) was not found in the <routine-type> specified. Floppy disk must be present as either the input or output device.

NO AVAILABLE SPACE ON MINIDISK

There is insufficient room on this floppy disk to complete the requested function. You should either use another floppy disk or obtain space on the current floppy disk by removing a file or files, or by purging the floppy disk.

NO CHANGED TO FILE-ID

A 'CH' function was requested but no output <file name> was provided.

NO FILES ON MINIDISK
FILE <file name> NOT LOADED

An image file could not be created as no files were present on the floppy disk.

NO INPUT FILE-ID PROVIDED

A file name must be supplied for the 'CH', 'RM' or 'RN' Analyze functions.

NO RECORDS IN INPUT FILE

The file to be copied to the floppy disk contains no records. This request will be aborted.

OUTPUT MINIDISK NAME IS REQUIRED

A flexible disk name must be input when converting an object code or S-code file directly onto a flexible disk.

VOL ID MISMATCH, MINIDISK ID <volume-ID>

The <volume-ID> of the floppy disk does not match the <multifile name> input to MDCOPV for the floppy disk.

WRONG FILE (NOT MCPP OR ASSEMBLER)

This object code file was not compiled using thye ASMBLR compiler, or was compiled with the BPL compiler but the "MCPP" dollar option was not set. This code file cannot be converted.

WRONG MINIDISK FORMAT
FILE <file name> NOT LOADED

This floppy disk cannot be used to make an image file.

1 TO 6 CHARACTERS FOR IMAGE-FILE NAME

The <image-file name> was missing or was greater than 6 characters in length.

SECTION 12
MCP MERGE INTRINSIC PROGRAM (MERG:)

MERG: is the only MCP merge intrinsic program you can use with ANSI-74 COBOL. Below is an overview of the MERG: program and a concise functional description. If you are interested in a complete discussion of this program, you should refer to the *V Series COBOL ANSI-74 Reference Manual.*

The MCP rolls the calling program out to disk and then loads the MERG: program into the same memory space as the calling program. After the data has been merged successfully, the MCP rolls the calling program back in, and the calling program resumes execution.

MERG: uses information from the input parameters to establish how much memory is required for the merge operation. First, one buffer is allocated for each input file; the buffer size is determined by the file record length times its blocking factor. If you have specified any alternate areas for a file in the COBOL program, then MERG: will allocate a second buffer for this file.

At this point, MERG: determines its own memory requirement. If if needs more memory than the COBOL program is utilizing, MERG: uses a GROW BCT to acquire more memory. This extra memory is returned to the MCP when control is returned to the COBOL program. If, on the other hand, MERG: finds that the COBOL program is larger than the calculated merge memory requirement, the MERG: uses this extra space to allocate a second buffer to any input file for which a second buffer has not already been allocated.

As stated above, two input file buffers will be allocated if required. However, two buffers are always allocated for the output data file.

Functional Description

The purpose of the MCP MERG: intrinsic program is to accept presorted input files and merge them together to create one output file. The MERG: can merge up to eight input files into a single output file. The physical block size of input and output files may be different. However, the files you want to merge must all have an identical record size.

Records are merged on the basis of the ascending or descending keys that you specify in the COBOL MERGE statement. The rules for comparing operands in a relation condition are used to compare corresponding key record fields of the input files being merged. When these key fields are equal, the order of the output of these records follows the order in which you named the files with the USING clause of the COBOL MERGE statement. All such "equal" records with one input file will be output before records from another file.

MERG: assumes that you have ordered all input records identically. If input records are not found in the sequence you specified with the merge keys, these improperly ordered records are output immediately after being read. Data records continue to be output from an out-of order file until a record is encountered that brings the file back into the proper sequence. This may cause the output to be in a different ascending or descending sequence than anticipated by the program that invoked the MERG:.

NOTE

No warning message is displayed when files are merged that contain incorrectly sequenced keys.

SECTION 13
DCP UTILITY PROGRAM (NIFMRG)

The NIFMRG utility enables a data communications subsystem to use more than one type of Data Communications Processor (DCP) at a time. The DCPs may be Unisys B 874s, B 974s, or DCDLPs. NIFMRG also enables a CP 2000 communication processor to function in a V Series data communication environment. The term, "DCDLP" refers both to an ORS DLP (offline Reader Sorter) and a TELCOM DLP. You may use multiple DCPs in a data communications subsystem without needing NIFMRG if all the DCPs are of the same type.

The NIFMRG program combines MCPNIF files and MCSNIF files to produce a single MCPNIF file and a single MCSNIF file containing information on all the DCPs. B 874 firmware files are also altered.

The NIF files to be merged are produced by the NDL compiler in the case of the B 874 and the B 974, and are supplied by Unisys in the case of the ORS DLP. One NDL compilation may include information on multiple DCPs of a single type, but not on more than one type of DCP. Refer to the appropriate NDL language manuals for details.

HOW TO OPERATE NIFMRG

To operate NIFMRG, two steps are required. First, you must initiate the program. Then enter parameter syntax input which directs its operation.

You can initiate NIFMRG from cards or from the operator display terminal (ODT). You may input parameter syntax from the ODT, from cards, or from a preexisting disk file.

1. If you intend to input parameter syntax from the ODT, you should initiate the program by entering the following from cards or the ODT:

   EX NIFMRG/SPO

   NIFMRG will then prompt input with the message:

   ===> ENTER NEXT STATEMENT <===

   You can enter parameter syntax by using AX keyboard input messages. Errors in syntax will be flagged, allowing correct syntax to be reentered. Input is free form; see "Statements Longer Than One Line", in the following paragraphs.

2. If you intend to input parameter syntax from cards, you should initiate the program by entering the following from cards or from the ODT:

   EX NIFMRG

   The program then looks for a card deck containing the parameter syntax input.

3. If you intend to input parameter syntax from a preexisting disk file, you should initiate the program by entering the following from cards or from the ODT:

   EX NIFMRG FILE INPARM=<disk file name> DSK

   The program will then look for a disk file with the specified name, containing the parameter syntax. The disk file must have 80 characters per record and one record for each block. You may create it through an editor or other means.

PARAMETER SYNTAX

There are four kinds of statements in the NIFMRG syntax: DCP, MERGED, END, and QUIT. DCP, MERGED, and END are required. You may enter QUIT only from the ODT; otherwise the syntax is the same whether you input it from cards, ODT, or disk file. Each statement ends with a period.

DCP Statement

This statement is required and is shown in Figure 13–1. You can repeat it any number of times, with each statement taking effect.

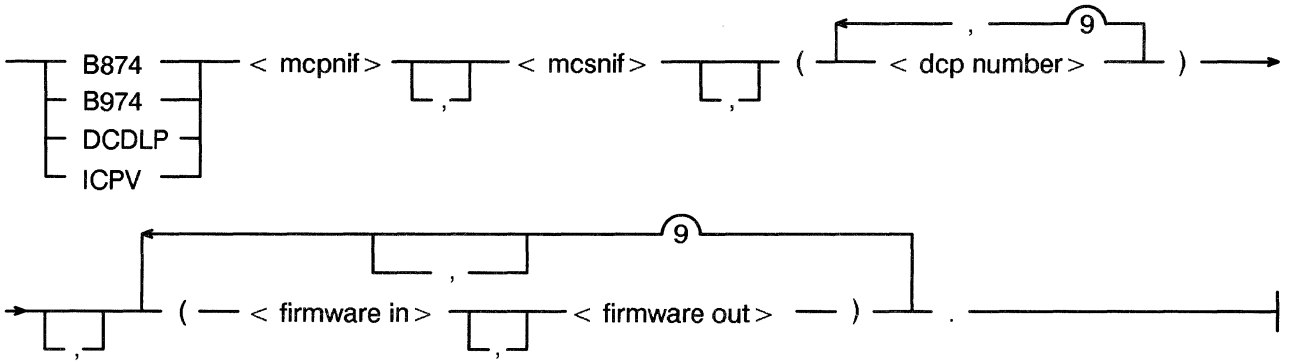


Figure 13-1. DCP Statement

- B874, B974, DCDLP, ICPV

  This specifies the type of DCP to which this DCP statement applies. DCDLP represents ORS-DLPs and TELCOM DLPs. ICPV represents a CP 2000 communications processor.

<mcpnif>
<mcsnif>

These are the file names of the MCPNIF and MCSNIF files as they were created by the NDL compiler or supplied by Unisys.

<dcp number>

These are the numbers of the DCPs to which this statement applies. The merged NIFs will contain information only for the DCPs that are declared. DCP numbers are assigned in the NDL source, which must be consistent with that specified on the DCP UNIT record at cold or warm start time.

<firmware in>
<firmware out>

<Firmware in> is the name of the firmware file as it was prepared by the NDL compiler or as supplied by Unisys. (The firmware file is named in the NDL "CODEFILE" statement; that name can be changed by an NDL compiler FILE-equate statement.)

<Firmware out> is the name of the firmware file as it will be prepared by NIFMRG. Because NIFMRG creates an output B874 firmware file which is a modified version of the input, it may be desirable to rename the output, thus ensuring that the unaltered original file is still present after the merge process, in case the merge is later repeated or abandoned. This is also the name placed in the MERGED MCPNIF for the firmware name of this DCP.

You must list the <firmware in> and <firmware out> file pairs in the same order as the DCP numbers are given. That is, if the DCP numbers are given in the order 4, 3, the first firmware in/firmware out file must pertain to DCP number 4 and the second to DCP number 3.

The name for the <firmware in> files are the exact names of the files as they currently exist. For example, assume that an NDL compilation contains the statement CODEFILE = MC9N0F in the DCP section, but that a compile-time FILE-equate renames the code file to CC9N0F. The FILE equate would cause the compiler to create a code file named CC9N0F. You would have to change that name to MC9N0F before installing the NDL, because it is the CODEFILE name which is actually stored in the MCPNIF file and is used by the MCP to identify the firmware file.

To merge the file using NIFMRG, the <firmware in> file name would be CC9N0F. The <firmware out> name could be different for example, it could be CX9N0F. In that case, NIFMRG would create a file named CX9N0F. The MCPNIF file created by NIFMRG, would specify CX9N0F as the firmware for the DCP.

Examples:

Ideally, there should be one DCP statement for each MCPNIF and MCSNIF in a network. However, you can use more than one DCP statement to add DCP numbers and firmware file names that were left out of a DCP statement, but be sure you do not specify the same DCP number twice. For example, if you entered the following statement, and two other DCPs were also declared in the NIF files:

B874 C8PN0F, C8SN0F, (1,2) (C8CDF1,F8WN0F)(C8CDF2,FIRO2).

The two forgotten DCPs could be added with the following DCP statement:

B874, C8PN0F, C8SN0F,(3,4) (CPCDF3,FIRO3)(CPCDF4,FIRO4).

MERGED Statement

This required statement names the newly merged NIF files. You can give any number of MERGED statements, but if you use more than one, an error is flagged and only the last statement takes effect. This allows you to change your mind about the file names. The syntax for the MERGED Statement is shown in Figure 13-2.

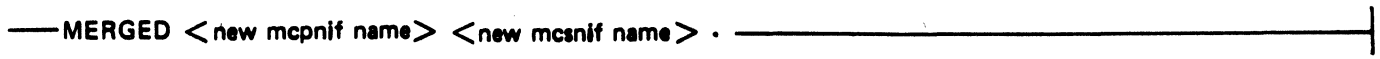


Figure 13-2. Merged Statement

Example:

MERGED CCPN0F CCSN0F .

If the newly merged NIF file has the same name as a file that already exists, a DUP LIB (duplicate library file) condition will arise. You can resolve this by entering a <mix number> RM keyboard input command, which will cause the old file to be overwritten by the new file.

This situation will not arise, however, if you set the MCP option RMOV. You can set this option on the MCP Confituration File, or at any time, by doing SO RMOV (set option).

END Statement

This must be the last statement of the input, since it signals the end of input to the NIFMRG program. If there are no errors in the syntax, NIFMRG will begin merging NIFs. If you are inputting syntax from the ODT, errors can be corrected and END entered again. The syntax for the END Statement is shown in Figure 13-3.

Note that the period is required.

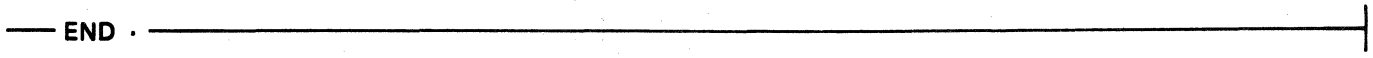


Figure 13-3. END Statement

Examples:

If B874 <firmware in> files are C8PN0F, C8SN0F, C8WN0F and B974 <firmware in> files are J9PN0F, J9SN0F, J99N0F then the NIFMRG input parameters are as follows:

```
B874 C8PN0F, C8SN0F,(1, 2), (C8WN0F,FC1N0F), (C8WN0F,FC2N0F).

B974 J9PN0F, J9SN0F,(9), (J99N0F,FC9N0F).

MERGED FCPN0F, FCSN0F.

END.
```

The <firmware out> files will be FCPN0F, FCSN0F, FC1N0F, FC2N0F and FC9N0F.

QUIT Statement

You can enter this statement only from the ODT. It ends the syntax and exits from the program, leaving all files untouched. Figure 13-4 shows the syntax for the QUIT Statement.

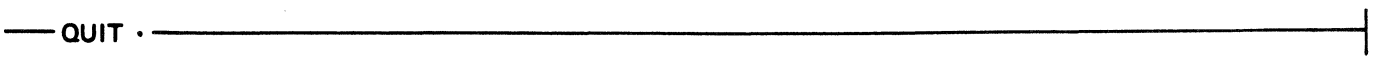


Figure 13-4. QUIT Statement

STATEMENTS LONGER THAN ONE LINE

There will be times when a DCP statement will be longer than the 80-column card image or the 60-column ODT line. All statements can occupy as many lines as needed; just remember to terminate each statement with a period. The following are examples of DCP statements longer than one line of input:

```
B874 C8PN1F C8SN1F (1,7)
(C81N1F,CC1N2F),(C87N1F,CC7N2F).
B974 C9PN5F C9SN5F
(5)
(C95N1F,C95N2F)
.
DCDLP CCPN8F,CCSN8F (3,4, 2),
(XX3N1F,XX3N2F),
(XX4N1F,XX4N2F),
(XX2N1F,XX2N2F).
```

ERRORS

When an error is encountered in the syntax, NIFMRG ignores all input until it finds a period. If the error is an omitted period, NIFMRG ignores all input until it finds another period. If input is from an ODT, NIFMRG will flag an error and corrected syntax can be entered.

LISTING

NIFMRG generates a listing. The first part includes the parameter syntax input, errors and where they occurred, and what NIFMRG took as a single statement. Statements are double spaced to show where one statement ends and the next begins. The lines for a single statement and its errors are single spaced.

The next part of the listing gives information about the MCPNIF and MCSNIF files. This table includes line numbers, for reference; the names of the original NIF files; the DCP type, the total number of DCPs and DCP numbers that the lines relate to. In the column labeled "DCP TYPE", 00 indicates B 874; 01 indicates B 974; 02 indicates DCDLP. A blank indicates an unused column.

A second table lists the DCP numbers with their related firmware in and firmware out file names.

The Global Information Count gives the number of valid entries that will be used from the first table and the names of the merged NIF files.

The last part of the listing tells how many DCPs are declared in the merged NIFs.

SECTION 14
PBDPRN UTILITY PROGRAM (PBDPRN)

PBDPRN is a bound utility that prints printer backup files on various types of printers, including Image Page Printers. This utility replaces PBDOUT and PBTOUT.

PBDPRN can print printer backup files by number or name. It can print these files from tape, and, conversely, create printer backup tapes. PBDPRN has additional features that let tell it to search the backup file for a certain string of text, and to begin printing when it finds the string. You can tell it to start printing at a certain position in a file, and to stop at another position. PBDPRN can also translate all lower case letters to upper case letters as it prints.

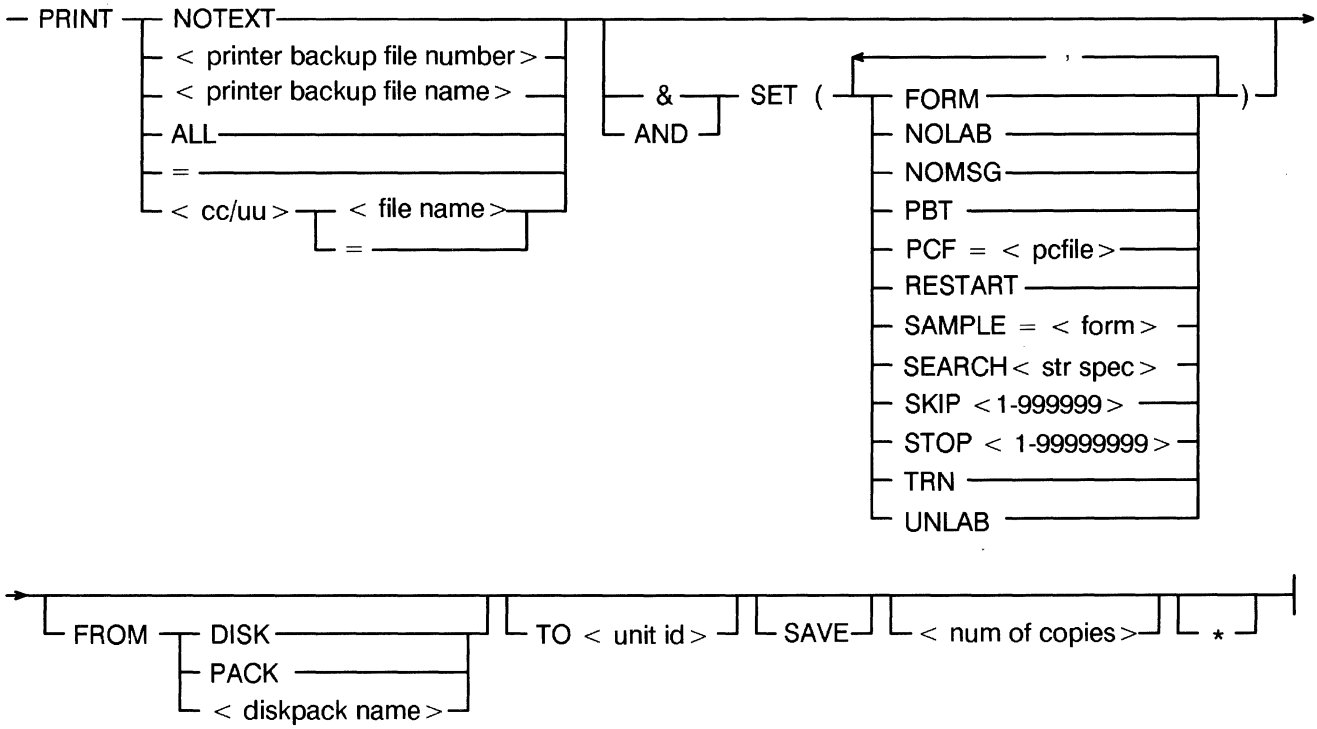


Figure 14-1. PRINT Syntax

PRINT

PRINT invokes the utility PBDPRN.

NOTEXT

You can use NOTEXT to print a sample of a form or a printer control file (PCF) produced by the Forms Manager program. It is used only with the Image Page Printer.

<printer backup file number>

PBDPRN prints individual files according to their backup file number.

<printer backup file name>

PBDPRN also prints individual files according to their backup file name.

ALL
=

All or the equal sign causes PBDPRN to print all printer backup files. Unless you specify DISK or PACK, this option prints all the printer backup files on both disk and diskpack.

cc/uu <file name>
cc/uu =

PBDPRN can print printer backup files from tape if you specify the tape drive's cc/uu number. If you use a file name, it prints only that file. If you enter an equal sign, it prints all the printer backup files on that tape.

AND SET
& SET

The options following the key words AND SET (or & SET, whichever you prefer) must be within parentheses. If you are using more than one option, each option must be separated with a comma.

FORM

FORM directs output to the printer previously named in the MCP file equate control card. For example, FILE PRINT = SHORT/OUTPUT.

NOLAB

If you use this option, a block letter label will not be printed on the cover page of a printout. However, the line that indicates the backup file number, its name and the file label information will be printed.

NOMSG

This option stops PBDPRN from displaying messages on the system console when it finishes printing a file.

PBT

PBT directs the printer backup listing to tape.

PCF = <file name>

This must be used to print backup files on an Image Page Printer. When NOTEXT is used, an entire print job (PCF), that can contain multiple forms, is printed as a sample.

RESTART

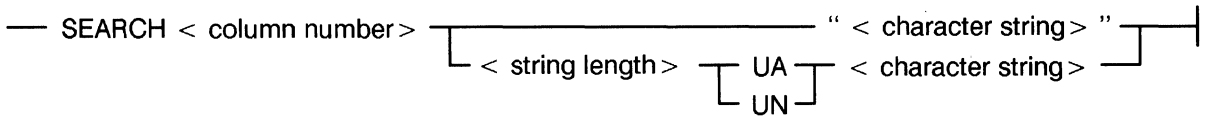
You can use this option to restart a print job that has been stopped with the MCP keyboard input message QT or as a result of some VSID message protocol errors that terminated PBDPRN. This option tells PBDPRN to start printing at the beginning of the page it was printing when it was stopped. If it does not find the beginning of a page, it resumes at the beginning of the last line it was printing when it was stopped. This option does not work for files on printer backup tapes (PBT).

To use this option, simply use the same syntax you used to start the job originally, but include the option RESTART.

SAMPLE = <form>

SAMPLE prints sample forms that are in the process of being created by the Forms Manager program. It prints only the form, no text. It is used only with an Image Page Printer and is reserved for use by the Forms Manager Program.

SEARCH <str spec>



SEARCH

This tells PBDPRN to search a column of text and to start printing the file where it finds a matching string of characters. The string can not exceed 99 characters.

<column number>

This tells PBDPRN which column to search. It can be any number between 1 and 132. Note, the sum of the column number and the string length can not exceed 133.

"<character string>"

This is the string of characters PBDPRN searches for. The string can not be longer than 99 characters, including blanks.

<string length>

This number indicates how many characters and blanks are in the string. There can be up to 99 characters in the string.

UA
UN

UA indicates the string is a representation of bytes as you would see on a printed listing.

UN strips the zone portions off a character. This is valid only for backup on tape condition, where you are printing a backup from DISK/PACK. There UA is assumed as the compare type.

SKIP <1-999999>

This option tells PBDPRN to begin printing at a certain line number. It can be used with the option STOP to print only part of a file.

STOP <1-99999999>

This option tells PBDPRN to stop printing at a certain line number. It can be used with the option SKIP to print only part of a file.

TRN

This option tells PBDPRN to translate all lower case letters to upper case letters.

UNLAB

If this option is used, no label or other types of information will be printed on the cover page of the printout.

FROM DISK
FROM PACK
FROM <diskpack name>

FROM DISK indicates a printer backup file on disk.

FROM PACK indicates a printer backup file on diskpack. PBDPRN searches:

- The primary backup family declared on the CONTROL BACKUP record during system initialization.
- The secondary backup family declared on the CONTROL BACKUP record during system initialization. (If no secondary backup family is declared, PBDPRN skips this search.)
- All unrestricted diskpacks.

FROM <diskpack name> indicates that the printer backup file is stored on a specific diskpack. PBDPRN searches only that diskpack for the file.

If no FROM clause is included in the PRINT command, PBDPRN searches disk, the primary backup family, the secondary backup family (if any), and all unrestricted diskpacks.

TO <unit ID>

Use this option if you want to use a particular printer. If you are using an Image Page Printer, you must use the option PCF = <file name>. If you use this option, the printer will be saved after printing.

If you do not use this option, PBDPRN prints the file on the first available line printer.

SAVE

If you use this option, PBDPRN will not remove the printer backup file after it is printed.

<number of copies>

Use this option if you want to print more than one copy. Up to 99 copies can be printed for each print job. This option is ignored if PBDPRN is printing backup files from a tape.

*

The asterisk tells PBDPRN to go into a waiting accept condition. PBDPRN will display ENTER SPECS on ODT and PROMPT you with ACCEPT. You can then AX parameters to PBDPRN with the MCP keyboard input message AX. The use and syntax of the parameters are the same as those used in PB keyboard command. Refer to the options available under PB keyboard command for detailed information.

Examples With Explanations

PRINT 12345

Print the printer backup file @12345 on any available line printer.

PRINT 12345 AND SET (SKIP 100, STOP 1000)

Print the printer backup file @12345 starting at line 100 and stop printing at line 1000. Because no printer was specified, print the file on any available line printer.

PRINT 6/1 =

Print all the backup files from the tape on drive with the cc/uu number 6/1 on any available line printer.

PRINT BKFILE & SET (SEARCH 10 "SAM") TO SPEC 5

Search column 10 of the text and begin printing when you find the string SAM. Print that entire line through to the end of the file. Use the printer SPEC. Print five copies.

PRINT 12345 AND SET (PCF=DOCPCF)

Print the file @12345 on the Image Page Printer.

PRINT NOTEXT & SET (PCF=FORMSS)

Print a sample of a form created by the Forms Manager program. Only print the form, no text.

Executing PBDPRN

PBDPRN can be executed with the MCP control instruction EXECUTE by using the following syntax:

EX PBDPRN.

After it is executed, PBDPRN sends the following command, requesting the command syntax be input with the keyboard input message AX:

PBDPRN = <mix number> ENTER SYNTAX. AX 'END' TO TERMINATE INPUT.

Auto Printing

MCP control file equation cards can be used to automatically direct printer backup files to the Image Page Printer.

The following examples illustrate the different syntax used to print automatically to a line printer versus an Image Page Printer.

To output to a line printer, the MCP FILE control card might look like:

?FILE LINE = SPEC/OUTPUT FORM AUTO

To output this file to an Image Page Printer, the card would look like this:

?FILE LINE = DOCPCF/OUTPUT FORM AUTO.

NOTE

DOCPCF is the unit ID for an Image Page Printer.

PBDPRN Syntax Errors

PBDPRN displays the following messages when there are errors in syntax input from a terminal.

AT <token>

This message indicates where in the syntax an error occurred.

BAD COMPARE STRING

Please refer to the PB keyboard input message.

BAD OFFSET

Please refer to the PB keyboard input message.

BAD SKIP FACTOR

Invalid skip detected.

BAD STRING LENGTH

Invalid length of compare string detected.

DESTINATION MISSING

The keyword "TO" must be followed by the <unit ID> of the printer PBDPRN is to print the file on.

EQUAL SIGN MISSING

The symbol = must be used with the PCF or SAMPLE options. For example, PRINT 12345 AND SET (PCF=MYPCF).

INVALID COPY NUMBER, TWO DIGIT NUMBER EXPECTED

PBDPRN prints a maximum of 99 copies for each execution.

INVALID FILE IDENTIFIER

File names can be up to six characters long.

INVALID OPTION, 'SET' EXPECTED

The keyword SET must be used when invoking the AND SET options. For example, PRINT 12345 & SET (NOLAB, TRN).

INVALID PCF PACK NAME

The name of a diskpack is up to six characters long.

INVALID SET OPTION OR ')' EXPECTED

Valid options are listed in the railroad diagram. They must be in parenthesis. For example, PRINT MYFILE AND SET (NOLAB)

INVALID SOURCE, 'PACK' OR 'DISK' EXPECTED

PACK OR DISK, are the proper names for media.

INVALID STOP NUMBER

PBDPRN stops printing files at a line number 1 to 99999999.

INVALID SYNTAX, 'FROM' OR 'TO' EXPECTED

Something other than FROM, TO, SAVE, copy number, * was specified after the file name and the AND SET options.

MISSING FILE IDENTIFIER

PBDPRN always needs to know what file to print. You can do this by specifying the backup file name, its number, or by using the keyword ALL or the symbol =.

MISSING LEFT PAREN

Parentheses are required with the AND SET options. For example, PRINT MYFILE AND SET (NOLAB).

MISSING PCF ID

The option PCF must be used with a file name. For example, PRINT MYFILE AND SET (PCF=DEFILE).

MISSING SAMPLE ID

A file name must be specified with the option SAMPLE.

'UA' OR 'UN' EXPECTED

When using the SEARCH option and specifying the length of the string, you must indicate if the string is UA or UN. An example of this syntax could be: PRINT 12345 & SET (SEARCH 1 3 UA SAM).

AX Errors

When an asterisk is used at the end of the command syntax, you can then input various options with the AX keyboard message. If there is an error in that input, PBDPRN displays one of the following messages.

BAD COMPARE STRING

Please refer to the PB keyboard input message.

BAD OFFSET

Please refer to the PB keyboard input message.

BAD SKIP FACTOR

Please refer to the PB keyboard input message.

ENTER SPECS

When PBDPRN is invoked with an asterisk at the end of the command string, it outputs this message, signaling that it is ready for options to be input with the MCP keyboard input message AX.

INVALID PARAMETER - TRY SOMETHING ELSE

The syntax entered is not correct. Try again.

Messages Related to the Image Page Printer and Line Printers

<F-N> PRINTED cc/uu COPY [<copy-number>]

Please refer to the PB keyboard input message.

<F-N> STOPPED cc/uu <copy-number> RECORD <record number>

Please refer to the PB keyboard input message.

***** PARITY ERROR AX OR DS?**

PBDPRN detected a parity error in a tape file. An AX forces PBDPRN to print error flags marking the corrupt block and to continue printing the file.

FILE NOT PRESENT

PBDPRN can not find the file on disk or diskpack.

FILE NOT STOPPED

Please refer to the PB keyboard input message.

INVALID FILE

Please refer to the PB keyboard input message.

INVALID PAPERMOTION FIELD
DS OR DP?

Please refer to the PB keyboard input message.

INVALID PRINTER TYPE FOR NOTEXT

The keyword NOTEXT is used only with Image Page Printers.

LINE PRINTER/IMAGE PRINTER

PBDPRN displays this message as it starts printing to indicate which printer it is in contact with.

NO MATCHING RECORD

Please refer to the PB keyboard input message.

PBDPRN =<mix number> CURRENT PRINT IS <percent> COMPLETE
ESTIMATED TIME TO COMPLETION IS hh:mm:ss (EOJ AT hh:mm)

PBDPRN issues this message, telling when it will finish printing a document. This message is in response to the keyboard input message SW. To find out when PBDPRN will finish printing, input <mix number> SW1 1.

PBK ON MORE THAN ONE PROCESSOR

More than one file with the same name exists on the shared system. PBDPRN will ask you for the processor number to be entered by AX command before continuing with the print job.

PRINT SKIPPED [FORWARD/BACKWARD] <a number of> RECORDS

A number of records have been skipped, forward or backward, as a result of the keyboard input message SK.

STRING FOUND AT RECORD <record number>

Please refer to the PB keyboard input message.

Messages Related to the Image Page Printer

BACKUP INFO SAVED

This indicates that printing terminated abnormally, and that the backup information has been saved. Resume the job at a later point in time by using the RESTART option. If the print job terminates and PBDPRN does not display message, the print job will have to be restarted from the beginning.

CHANNEL <channel number> NOT SPECIFIED IN PCF
AX or DS?

AX forces PBDPRN to begin printing at the top of the next page. DS discontinues the job.

FAILED AT REVERSE READ

PBDPRN has encountered an irrecoverable error.

FAILED READ FORWARD

PBDPRN has encountered an irrecoverable error.

FORM NOT FOUND IN THE FILE REQUESTED

PBDPRN can not find a form in the form file specified by the Image Page Printer. The program will terminate.

GRAPH IMAGE NOT FOUND IN IMAGE FILE

PBDPRN can not find a graphic image in the image file specified by the Image Page Printer. The program will terminate.

IMAGE NOT FOUND IN IMAGE FILE

PBDPRN can not find a specific font image in the image file specified by the Image Page Printer. The program will terminate.

INVALID PCF, FAILED AT FIRST READ

PBDPRN was unable to read the first record from PCF file that is a header record.

INVALID PCF, UNMATCHED PCF HEADER

The header does not agree with PCF format or the soft versions do not agree with MCP version.

IPP LOAD HOST REQUIRED
IPP DLP ERROR
IPP TRANSFERRED TO IDLE STATE UNEXPECTEDLY
INVALID VSID MSG FROM IPP

The MCP detected an error and is terminating the program. The MCP will schedule a Load Host for the Image Page Printer, if required.

INVALID SAMPLE FORM, PAGE DEF > 1

Only one physical page definition allowed for each SAMPLE form

MORE THAN ONE LOGICAL PAGE DEF

PBDPRN supports only one logical page definition for each physical page definition.

MORE THAN 20 DIFF FONTS ON ONE PAGE

There can not be more than 20 fonts for each logical page.

PAGE DEFINITIONS EXCEED 100 PER PCF

A maximum of 100 physical page definitions allowed for a PCF.

PAGE WANTED NOT IN QUEUE

A circular queue is maintained in PBDPRN corresponding to the circular buffer in IPP. The pages sent to IPP but not yet printed are remembered here. The IPP needs an error recovery action. Some pages need to be retransmitted, but the required page is not in the queue of PBDPRN. Recovery is impossible.

PCF UNEXPECTEDLY ENDS

PBDPRN failed to read a physical page definition from a printer control file.

STRING EXCEEDS 6 CHARS

The external file name of an image file or a form file can be up to 6 characters long.

STRING EXCEEDS 48 CHARS

The names of the fonts and forms can not be longer than 48 characters.

UNEXPECTED VSID STATE

PBDPRN detected an invalid Image Page Printer state and can not continue printing. Any unprinted material has been saved, and the job can be restarted with the option RESTART. If PBDPRN was printing from a backup file on tape, the job is abandoned. In this case, the backup key is not saved in a file, but is displayed on the ODT. This backup key can be used with the option SKIP at a later point in time to print the remaining part of the file.

UNIDENTIFIED TABLE ID IN GET PARTIAL TABLE

PBDPRN can not identify what table the Image Page Printer needs. The program terminates processing.

Image Page Printer Error Recovery Messages

If the Image Page Printer detects an error in the data sent to it by PBDPRN, it may display one of the following messages. In some cases, the Image Page Printer expects one of several responses to be input. Each message outlines its own options.

DATA STRUCTURES ERROR
<message from Image Page Printer>
ENTER OPTION #, 1:ABORT DOCU 2:ABORT PAGE

The Image Page Printer found an error in the data it is processing. You must enter one of the options with the MCP keyboard input message AX. For example, you might enter the option like this: <mix number> AX 1.

REJECT HOST MSG
<reason>
PRESENT STATE: <IPP state>, NEXT STATE: <state IPP requires>
SYNC TO NEXT STATE INITIATED

The Image Page Printer has encountered an error. The <reason> for the error is sent by the Image Printer. The present state or next state are IPP generated, and PBDPRN tries automatically to sync to the NEXT state.

INVALID OPTION, TRY AGAIN

When entering options in response to the recovery messages USER DATA ERROR, UNAVAILABLE FEATURE or DATA STRUCTURE ERROR, AX the number of the response, not the word. Moreover, only AX the numbers 1, 2 or 3. In some cases, there are only two responses.

ERROR RECOVERY BEGINS -
<messages from IPP>

PBDPRN displays this message after a response number 2, ABORT PAGE. The subsequent <messages from the IPP> appear on the ODT result from efforts to clear out the output queue, of the Image Printer, so it can go about the business of the ABORT PAGE.

VSID STATE MATCHED! ERROR RECOVERY ENDS

This message indicates that the recovery for an ABORT PAGE, option 2, was successful.

IPP NOT IN PROPER STATE! PROGRAM TERMINATING

This message indicates that the recovery for an ABORT PAGE, option 2, was not successful. The program terminates.

REJECT HOST MSG
<message from Image Page Printer>
PRESENT STATE: <IPP state>, NEXT STATE: <IPP state>
[SYNCHRONIZATION REJECTED! PROGRAM TERMINATING]

While in error recovery, the attempt to sync to the next state failed. The program terminates.

RESTART PAGE, PAGE: <page number>

One or more of the pages sent to the Image Printer were lost before they were printed successfully. All the pages beginning with <page number> must be retransmitted to the printer. PBDPRN will attempt to perform all necessary recovery action.

UNAVAILABLE FEATURE!
<message from Image Page Printer>
ENTER OPTION: 1 = ABORT DOCU, 2 = ABORT PAGE [,3 = IGNORE]

A feature has been invoked that is not available on the printer, such as duplex printing, scaling, rotation, and so forth. The program expects one of the options to be input with the MCP keyboard input message AX. Inputting 1 aborts the printing of the document; Inputting 2 aborts the printing of the page; Inputting 3 makes the program ignore the error. Note, in some cases, option 3, IGNORE, would not be a valid option and is not displayed.

USER DATA ERROR!
<message from Image Page Printer>
ENTER OPTION #, 1:ABORT DOCU, 2:ABORT PAGE, 3:IGNORE

The printer has detected a programmer or form definer logical error. Use the MCP keyboard message AX to input the number of the proper response. The options are explained in the message UNAVAILABLE FEATURE above.

SECTION 15
DISKPACK COPY UTILITY (PKCOPY)

Use PKCOPY to duplicate all the data on one diskpack on another diskpack, making an exact duplicate of the first. Both diskpacks must be of the same type and model. The copy is performed on sector by sector, direct I/O basis, which gives it some speed advantage over SYSTEM/COPY. Again, the packs must be of the same type and model. Moreover, the destination pack should be formatted before it is used.

This utility has a feature that compares the data as it is copied to the diskpack. If it detects a discrepancy, the utility displays a message and gives you the option to stop the copying process or to continue. It also has a feature that lets you change the serial number of the destination diskpack. Otherwise, both diskpacks will end up with the same serial numbers. You should not change the serial numbers if the source pack contains parts of a file that span two or more diskpacks. In such a case, the diskpack serial numbers are a vital source to finding all the parts of the file.

When the copy completes, one diskpack becomes an exact duplicate of the other. They will have the same files, the same family name, the same *available* table, and, depending on whether or not you use the feature, the same serial numbers.

The MCP uses the available table to keep track of what space is available on the diskpack, and to omit bad sectors from use. The system command XP is used to indicate what diskpack sectors are bad and cannot be used. The information input with the XP command is stored in the available table. Therefore, the duplicate diskpack, which has a duplicate available table, has the same sectors omitted from use as the source pack does.

Nothing can access either diskpack until the copy completes. When it does complete, both diskpacks end up in a saved mode. When a diskpack is in the saved mode, it cannot be used until it is made available with the system command RY.

The default amount of memory allocated to the utility when it executes is 42KD. The default buffer size is 36KD. One hundred sectors are used for each I/O for copying. Fifty sectors are used for a COMPARE. For larger I/Os, additional memory can be provided with the system command MEMORY.

Operating Instructions

Operating PKCOPY is relatively simple. First you execute it, and then you input PKCOPY commands to it. PKCOPY is executed with the command EX. PKCOPY commands are then input with the command AX.

Use the following command to execute PKCOPY:

```
EX PKCOPY
```

After you execute the utility, get its mix number. You need the mix number in order to input the following commands to the utility. Use the following syntax to pass commands to PKCOPY:

Figure 15-1. PKCOPY Syntax

<mix number> AX

This is the command that passes information to PKCOPY. The mix number is the mix number issued to PKCOPY after it is executed. Be sure to use the right mix number. Note that quotes enclose the information being passed to the utility.

"COPY

This keyword begins the PKCOPY command syntax. Note that it must begin with a quote.

COMPARE
AND COMPARE

Either of these invoke a feature that compares the data on both packs. If there is a discrepancy, PKCOPY displays an error message, and a prompt asking if you want to continue.

<source diskpack cc/uu>

This indicates the channel and unit numbers of the diskpack the data is to be copied from.

<destination diskpack cc/uu>

This indicates the channel and unit numbers of the diskpack the data is to be copied to.

<serial no.>

Unless you specify a serial number here, both diskpacks end up with the same serial number. Diskpack serial number can range from 1 to 999999. Do not change the serial numbers if the diskpack contains parts of a file that spans two or more diskpacks. In such a case, the serial numbers are a vital source to finding all the parts of the file.

Examples:

The following examples have two parts. The first part is the command that executes the utility. The second part is the command input to the utility after it has been executed.

```
EX PKCOPY

26 AX "COPY 04/02 04/03"

EX PKCOPY

26 AX "COPY COMPARE 14/02 04/03 206043"

EX PKCOPY

26 AX "COPY AND COMPARE 04/02 04/03 206043"
```

PKCOPY Messages

PKCOPY displays messages when it encounters errors. When it encounters errors while reading or writing, it displays a message asking whether or not you want to continue.

"DO YOU WANT TO CONTINUE? (YES/NO)"

After this prompt is displayed, use the <mix number> AX command to input YES or NO. For example, if you to continue, you might enter the following command:

```
26 AX YES
```

The following paragraphs document PKCOPY error messages:

COMPARE ERROR ON SECTOR <sector address>

PKCOPY copies data on a sector by sector basis. In this case, a compare operation discovered a discrepancy between two sectors which should contain the same data.

INPUT/OUTPUT DRIVE INVALID OR BUSY

The copy cannot continue because of one of two reasons: either one of the two diskpacks is in use, or the channel and unit numbers you specified for one or both of the diskpacks are not correct.

INVALID FUNCTION - USE 'COPY' OR 'COPY [AND] COMPARE'

One more of the commands input after the AX is not spelled right, or is invalid.

INVALID PACK LABEL <name>

The utility did not find the diskpack label <name>.

NO COPY - PACKS MUST BE OF IDENTICAL TYPE

PKCOPY copies data between the same types of diskpacks. It does not copy data from one type of diskpack to another type of diskpack. You must use the utility SYSTEM/COPY to do that.

NO COPY - UNKNOWN UNIT ID RECEIVED

The unit number specified for one of the diskpacks is not correct.

READ ERROR ON SECTOR <sector number> UNREADABLE RD = <result descriptor>

A read operation from the indicated diskpack sector did not complete correctly. This may indicate a bad sector, in which case the sector may contain corrupt data.

WRITE ERROR ON PACK <cc/uu> RD = <result descriptor>

A write operation to the indicated diskpack sector did not complete correctly. This may indicate a bad sector, in which case the data it contains may be corrupt.

SECTION 16
SHARED SYSTEMS AND DEVICES (SHARED)

Shared systems provide a method for multiple programs (including the MCP) to concurrently access disk and diskpack files. In addition, through this concept, multiple systems (up to four) can concurrently access files resident on shared disk or diskpack media.

Specifically, a shared system accomplishes some or all of the following goals:

- Allow multiple programs to concurrently access a common file. Additional I/O operations and a Block Lockout Table (BLT) are provided to allow block level locking to occur to prevent simultaneous updating. This is best illustrated by the following scenario.

  a. Program A reads a record of a file.

  b. Program B reads the same record.

  c. Program A modifies some fields and writes the record.

  d. Program B modifies some fields and writes the same record, thus destroying the update accomplished by Program A. To prevent this, each program uses shared I/O operations to lock out other programs while it updates the record. Here is another example.

     1) Program A reads a record using a read with lock operation.

     2) Program B tries to read the same record in the same file but cannot because program A has already locked the record. Program B must wait until program A unlocks the record.

     3) Program A modifies some fields, writes the record, and then unlocks it.

     4) Program B can now lock and read the record, update some fields and then write the record and unlock it.

- Allow multiple systems to share disk and diskpack resources. This is accomplished through additional hardware called a Shared Systems Processor (referred to an SSP), which acts as a common repository for all locks and contentions between the shared systems.

- Provide a growth path for a memory bound environment. Hardware memory limitations are often overcome by the addition of another mainframe tied to the existing peripherals to provide added memory resources to the user environment.

- Provide a growth path for a processor bound environment. Again, the addition of another mainframe tied into the existing peripheral network is used to augment the processing power of the user environment.

- Provide redundancy. Many users require a fault tolerant environment. Shared systems allow the duplication of mainframe resources (system, memory, DLP's, and so forth) either in a passive or active state. Often, one system is dedicated to on line activity while the other is processing batch mode. If the critical on line system should fail, the batch system can take over that responsibility with minimal disruption.

Concurrent file access is regulated by a special set of file I/O operations. By using shared file I/O operations, programs can read, seek, and write file blocks in a manner that prevents data corruption by other programs that are also using shared file I/O operations. Note that if a program opens a file and

accesses it without using shared file I/O operations, the file is not protected and data corruption can result. The following rules apply to shared file I/O operations:

- If one program opens a file for shared access, all programs should open the file shared. Locking and unlocking protocols should be adhered to to prevent simultaneous updates.
- All programs must declare the same record size and blocking factor for the file they are sharing.
- Shared files must be declared random access.
- A program may re-lock a record (e.g., LOCK followed by READ LOCK)
- A program shall be terminated if it attempts to unlock a block that it does not have locked.
- A program shall be terminated if it attempts to write to a block that is locked by another program. Some locking-type operations (e.g., LOCK, READ LOCK, and so forth) should be done prior to the write.

Note that a program performs lock/unlock operations on a record level, but since the MCP deals with blocks of records for I/O, this locking is actually accomplished on the block level. This is what imposes the restriction that all accessing programs must reference a shared file with the same record size and blocking factor.

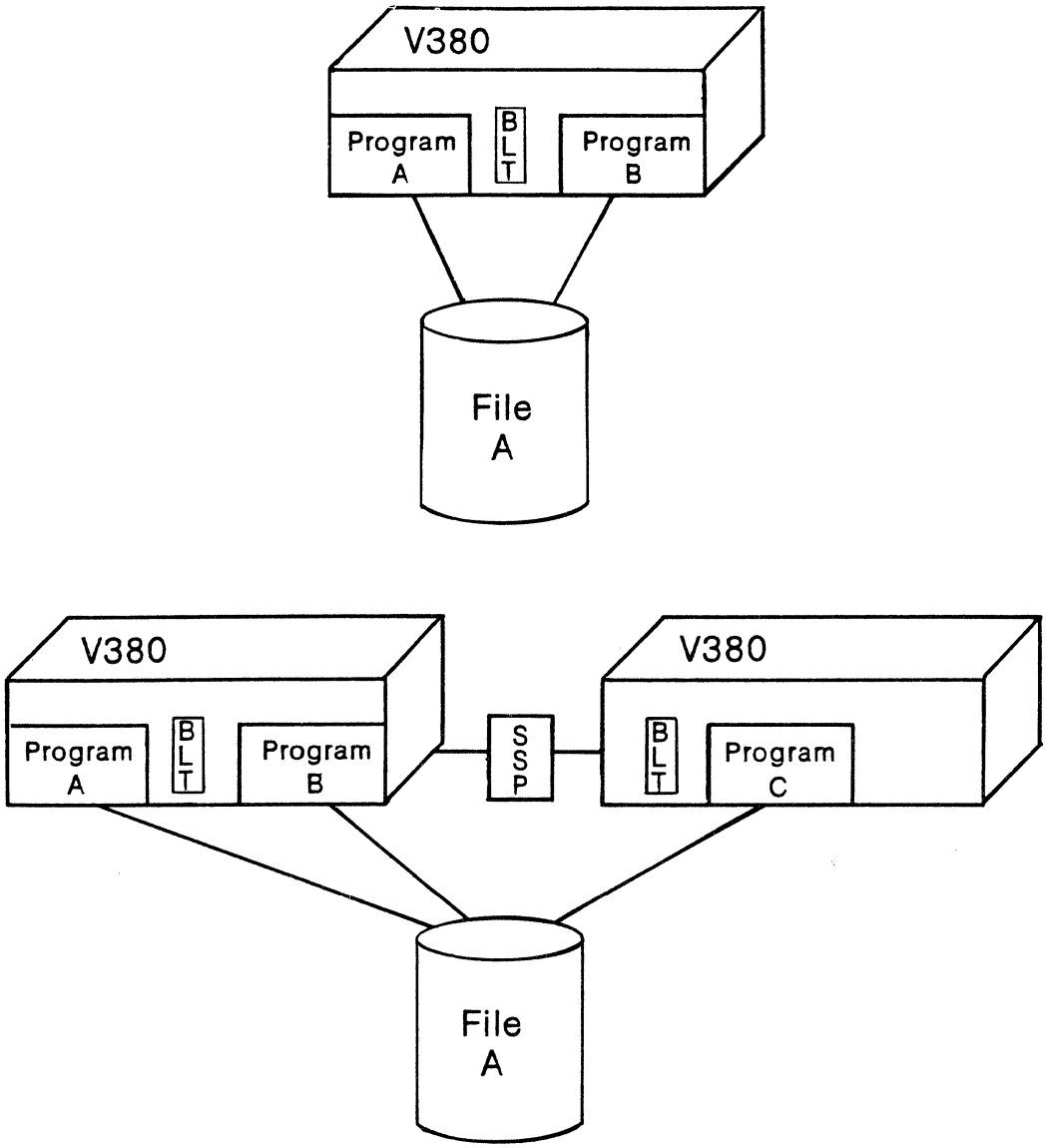


Figure 16-1. Concurrent File Access

Only the COBOL and BPL programming languages support shared file I/O operations. The types of shared file I/O operations and the corresponding syntax appear in Table 16-1 in alphabetical order.

Table 16-1. I/O OPERATIONS

FUNCTION	BPL	COBOL	BCT/VARIANT
LOCK	LOCK	READLOCKONLY	0114/4
LOCK NO CONTEND	LOCK SEEK	SEEKLOCKONLY	0314/4
UNLOCK	UNLOCK	UNLOCK	0234/8
READ	READ	READ	0114/0
LOCK, READ	READ LOCK	READLOCK	0114/8
LOCK, READ, UNLOCK	READ NO LOCK	READLOCKWAIT	0114/C
SEEK	SEEK	SEEK	0314/0
LOCK SEEK	SEEK LOCK	SEEK LOCK	0314/8
LOCK, SEEK, UNLOCK	SEEK NO UNLOCK	SEEKLOCKWAIT	0314/C
WRITE	WRITE LOCK	WRITELOCK	0234/4
WRITE, UNLOCK	WRITE	WRITE	0234/0

Single-System Shared Environment

A single shared system does not need any hardware modification to perform concurrent file access. All disk and diskpack media are capable of storing shared files and the system runs shared by default. An SSP is not required in a single system shared configuration.

All programs sharing a file must declare the same blocking factor for the file. When you access the file you must declare it random access and shared.

Multisystem Shared System Environment

In a multi-system shared environment, you declare and name an SSP with the DLP configuration record and all disks and diskpacks that the multiple systems will access should be set up as SHARED VIA <SSP name>. Not all media must be shared. Keep the following shared configuration rules in mind:

- If any disk is shared, subsystem 1 must be shared because the MCP and disk directory must be available to all systems.
- Additional disks may be shared, but all disks belonging to a specific subsystem must have the same shared attributes.
- QWIKDISK cannot be shared, since it is not accessible by other systems.
- All diskpacks belonging to the same family must have the same shared attributes, either shared or non-shared.

Because the MCP considers all disk and diskpack media eligible to store shared files by default, file placement can no longer be guaranteed. Thus, all the SHARED open attribute does is enable the program to issue shared file I/O operations to that file. This is different from previous MCPs, which only allowed shared file assignment to media declared SHARED.

All programs accessing a shared file must do so consistently. That is, the same file could (erroneously) be opened simultaneously shared and non-shared. This can lead to the loss of the record protection that shared is intended to afford. This is consistent with previous MCPs.

Types of Shared Systems

The following paragraphs illustrate and describe various types of shared system configurations.

Shared System

"Shared" refers to the ability of programs to access the same disk or diskpack file simultaneously, yet be afforded protection from simultaneous update through a shared record locking/unlocking protocol.

Single-System Shared

This refers to a shared system that uses only one mainframe. Multiple programs will be able to simultaneously access a common file. Under previous MCPs, single-system shared configuration was invoked by the SHARED option of the DSKn and PACK MCP configuration records, and the SHRD system option. MCP/VS 2.0 eliminates these requirements and the system is considered to be single system- shared by default.

MultiSystem Shared

This refers to a shared system that is extended to multiple mainframes.

A multi-system shared configuration requires the addition of a Shared Systems Processor (SSP) to handle the locking/unlocking between systems. Up to four V Series mainframes may be configured to form such a system.

Shared Disk Only System

A multisystem shared configuration that has one or more disks as common resources and no shared diskpack resources. The MCP disk must be shared. Each system runs the same MCP code file and shares the disk directory and available table. In this configuration, subsystem 1 must be shared, and additional disks can be shared on a subsystem level.

Shared Diskpack Only System

A multiple system configuration that has one or more diskpacks as common resources and no shared disk resources. The peculiarity here is that each system is running a separate copy of the MCP (albeit the same version and ASR).

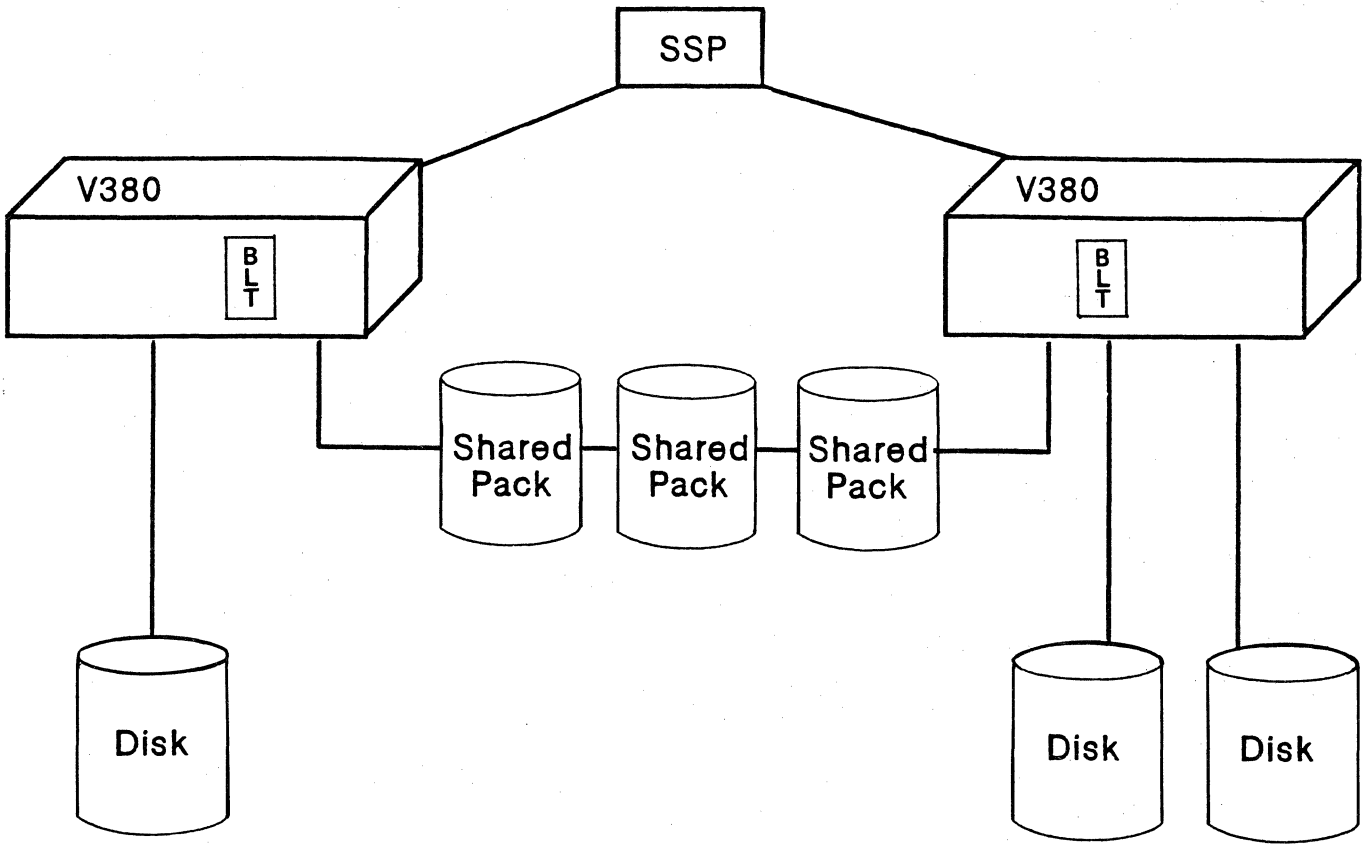


Figure 16-2. Shared Diskpack Only System

Shared Disk And DiskPack System

A multisystem shared configuration that has one or more disks and one of more diskpacks as common resources. The MCP disk must be shared. Each system runs the same MCP code file and shared the disk directory and available table. Additional disks may be shared at the subsystem level, while diskpacks may be shared at the family level.

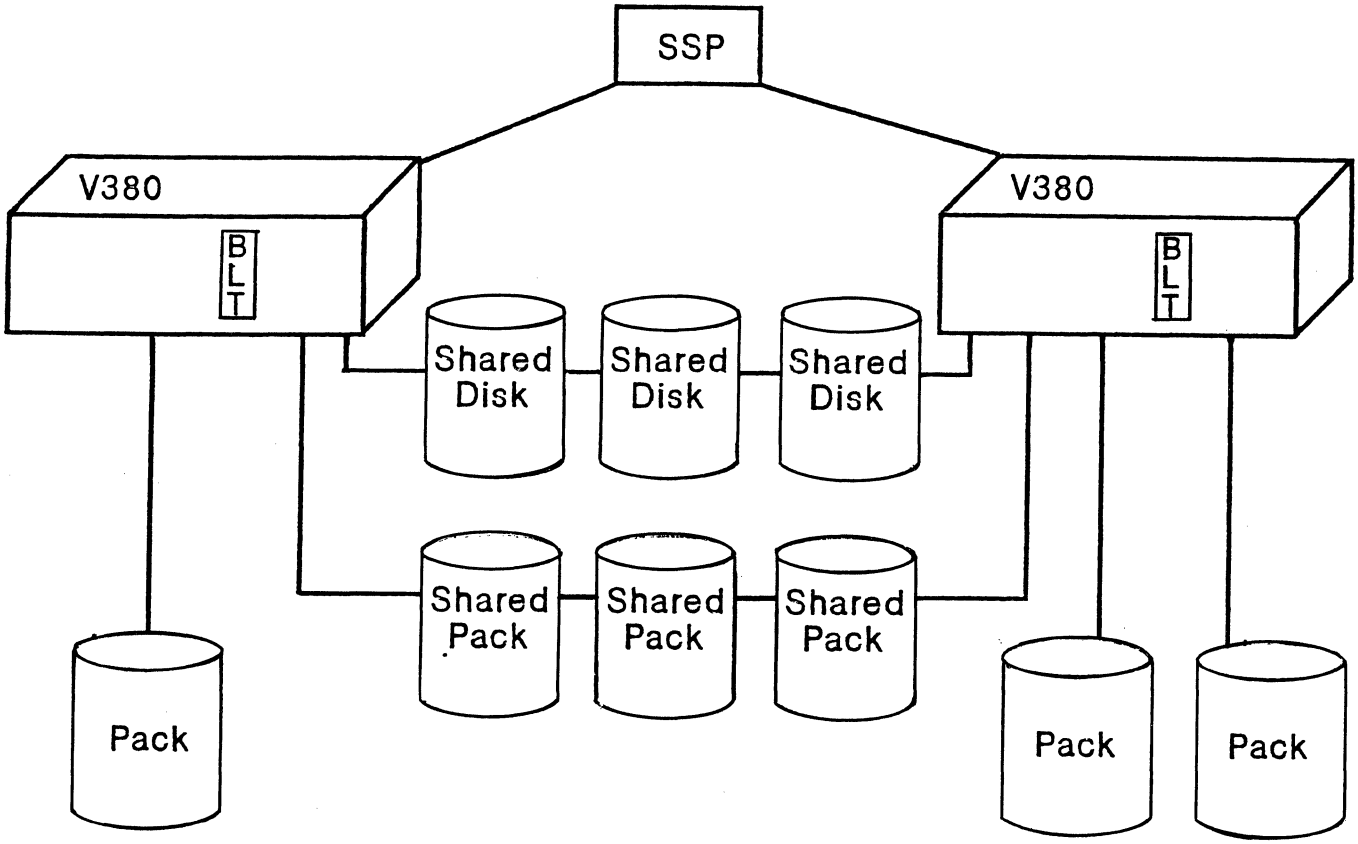


Figure 16-3. Shared Disk and Diskpack System

Definitions of Shared File I/O Operations And Other Related Terms

The following paragraphs define the shared file I/O operations that appear in Table 16-1 as well as other related terms.

BLT Contention BLT contention occurs when one program wants to access a file block that another program running on the same system has locked. See the portion of this section that covers the Block Lockout Table (BLT). When the programs are running on different systems, this situation is called SSP contention.

Contention When one or more programs attempt to lock a block that is locked by another program, they are suspended and must wait until the block is unlocked. The suspended program is said to be contending for the block.
The amount of time a program can contend for a file block is called the contention cycle. If the block does not become available during the contention cycle, the operating system intervenes and either reinstates the program at a stalemate use routine, if one has been coded into the program, or terminates the program.
The default contention cycle is 10 seconds. You may use the LIMIT DELAY record in the MCP configuration file to override the default. See the *V Series MCP/VS System Software Operations Guide, Volume 1: Installation* for more information about this record.
If two or more programs are waiting in contention, the program with the highest system priority gets the block first. You use the system command PRIORITY to set the program's priority. The system then restarts the contention cycle for the other programs which are contending for the block.

Contention Cycle The contention cycle is the period of time that a program can contend for a file block. After this period of time expires, the system either reinstates the program at a stalemate use routine, if one has been coded into the program, or else it terminates it. See *Contention* in the previous paragraphs for more information.

LOCK The LOCK shared file I/O operation locks a block and prevents other programs that are using shared file I/O operations from locking the block. The block remains locked until it is unlocked. If the block is locked by another program, the program will contend for the block.

LOCK NO CONTEND The LOCK NO CONTEND shared file I/O operation performs the function of the normal lock operation but will not contend for the lock should it be locked by another program. In this case, the MCP would immediately reinstate the program at a stalemate use routine, if one has been coded into the program, or terminates the program.

LOCK READ The LOCK READ shared file I/O operation locks a block and then reads it. The block remains locked until it is unlocked.

LOCK READ UNLOCK The LOCK READ UNLOCK shared file I/O operation locks a block, reads it, and then unlocks it.

LOCK SEEK The LOCK SEEK shared file I/O operation locks a block and reads it to a buffer for later use. The block remains locked until it is unlocked.

LOCK SEEK UNLOCK The LOCK SEEK UNLOCK shared file I/O operation locks a block, reads it to a buffer for later use, and then unlocks it.

MAILBOX An intersystem message-passing mechanism used to synchronize systems for MCP functions that require exclusive shared resource access. For MCPVS 2.0, the mailbox has been moved into the SSP, and is no longer on disk. The MCP configuration record that declared a MAILBOX for previous MCPs is not used with MCP/VS 2.0. However, the DLP SSP record in the MCP configuration file must include the MAILBOX attribute for the SSP used that contains the mailbox.

READ The READ I/O operation reads a file block, even if another program has locked that block.

RLT This refers to Record Lockout Table, the MCPIX/MCP/VS Mark 1.0 version of the MCP/VS Mark 2.0 Block Lockout Table (BLT).

Stalemate Stalemate is the term for a situation in which two or more programs cannot continue

processing because they are trying to lock mutually exclusive blocks within a file. Because they are waiting for each other, their contention cycle expires and the system either reinstates them at a stalemate use routine, if one has been coded into the program, or else it terminates them.
Here is an example of a stalemate:

1. Program A locks record 10 of a file.
2. Program B locks record 20 of the same file.
3. Program A attempts to lock record 20 of the same file, fails, and goes into a contention condition.
4. Program B attempts to lock record 10 of the same file, fails, and goes into a contention condition.
5. The period of time these programs can wait in contention (10 seconds or as set by the LIMIT DELAY record) expires. The MCP intervenes and either terminates the programs, or else it reinstates them at a stalemate use routine.

Note that the MCP does not detect true stalemate conditions. It merely sets a limit to the time that a program can wait for a lock without being given an opportunity to retry it (by another program unlocking). The reason for this lack of opportunity can range from the program which owns the lock not being given the sufficient processor resources perform an unlock, to a true stalemate condition as illustrated above.

SSP Contention SSP contention occurs when a program running on one system wants to lock a block that a program running on another system has locked. If the programs are running on the same system, it is called BLT contention. Also see the entries in this section that cover the *contention cycle* and the *Shared System Processor (SSP).*

Stalemate Use Routine The compilers and the MCP provide a mechanism to reinstate a program at a routine instead of terminating it when contends for a period of time exceeding the contention overtime cycle. The sophistication of the stalemate use routine can vary from simply ignoring the condition and returning to the main program logic, to backing out a previous series of updates and allowing the other programs to complete their operation.

SEEK The SEEK I/O operation locates a block and reads it to a buffer for later use, even if another program has locked that block.

UNLOCK The UNLOCK shared file I/O operation unlocks a locked block. If the program that uses this operation does not already have the block locked, it is terminated.

WRITE A WRITE I/O operation expects the relevant block to be already locked. It will, however, attempt the lock if it is not. Should this attempt fail, the program will be terminated. If the block was already locked by the program. or if the attempted lock were successful, the block will be written and then unlocked.

WRITE NO UNLOCK Whereas the WRITE operation is somewhat forgiving in the prelock requirements, this WRITE variation requires that the block be already locked by the program, or the program will be terminated. Additionally, once the block is written, it is left locked. Note that these WRITE operations and requirements are consistent with previous MCPs.

COMPONENTS OF A SHARED SYSTEM

The following paragraphs define and discuss the components of a shared system:

BLOCK LOCKOUT TABLE (BLT) The BLOCK LOCKOUT TABLE (BLT) keeps track of what file blocks are locked by which programs. The BLT resides in memory and the MCP updates it every time a program locks or unlocks a file block. The MCP provides a BLT with a sufficient number of entries to handle most circumstances. However, this default setting can be changed with the MCP configuration record LIMIT BLT. Refer to the LIMIT BLT record in the *V Series System MCP/VS Software Operations Guide Volume 1: Installation* for more information.

DISK When viewed from the single system perspective, all disk is considered eligible for shared file access. From a multisystem perspective, disk sharing is determined at the subsystem level. This means that all members of the same disk subsystem must have the same shared attributes. The SHARED option on the MCP configuration record DISK declares a disk to be multisystem shared. Refer to the *V Series MCP/VS System Software Operations Guide Volume 1: Installation.*

DISKPACK When viewed from the single system perspective, all diskpack is considered eligible for shared file access. From a multisystem perspective, diskpack sharing is determined at the family level. You can declare some diskpack units shared and you can declare other diskpack units not shared. All members of the same diskpack family should be either shared or non-shared.
The SHARED option of the MCP configuration record PACK declares a diskpack to be multisystem shared. When you declare a diskpack shared, you must assign the same logical ID number and the same SSP name for each system.

FAMILY NAME Individual diskpacks can be organized into families (groups). A "family name" is the name given to a group of diskpacks that is organized into a family. The MCP can then direct individual files to and store individual files on these groups of diskpacks according to family name.

LOGICAL SUBSYSTEM A logical subsystem represents a "family" of disk devices. As diskpack devices may be grouped into families for file assingment, disks are similarly ordered into subsystems. Subsystems have attributes such as default (available for generic file assignments) and shared (accessible to multiple systems). A disk's subsystem is defined by the SUBSYSTEM option of the MCP configuration record DISK.

MASTER CLEAR Master Clear is an electronic signal that the system uses to initialize a DLP and its associated hardware. Master Clear does not clear the SSP. When you do a BASE CLEAR, make sure that none of the systems attached to the base are using the SSP since it will cease to function and have to be reloaded.
When you enter TERM on the V Series, you generate a MASTER CLEAR. You generate BASE CLEAR with the switch on the base.

PHYSICAL SUBSYSTEM A physical subsystem is a group of disks and diskpacks or both connected to the system via the same DLP.

SHARED SYSTEM PROCESSOR (SSP) A Shared system processor (SSP) is a special hardware device that controls concurrent file access by programs running on two to four V Series systems. It is a specialized four card DLP which resides in a dedicated base shared by the systems. The SSP keeps track of the file locks held by a system, and which systems may be contending for those locks. It also stores the intersystem mailbox.
The SSP must be loaded with the proper firmware before it may be used. MCP/VS Mark 2.0 requires firmware level SSP302 or greater. Note also that the SSP must never be reloaded while in use by other systems since all locks and contentions would be purged. See the section on LOADFW in this manual for more information about how to load firmware to SSPs.
For more information, see the MCP configuration record SSP DLP. Refer to the *V Series MCP/VS System Software Operations Guide Volume 1: Installation.*

CONFIGURATION AND INITIALIZATION OF SHARED SYSTEMS

The following paragraphs discuss the configuration and initialization of single and multi-system shared systems.

Single System Shared

Initialization of a single system shared system is not that different from a normal system initialization, since the system is designed to support this feature by default. Note that this differs from previous MCPs,

where disk and diskpacks used to store shared files were declared shared. The only modifications may be to increase or decrease the number of BLT entries or the contention overtime cycle via the LIMIT BLT and LIMIT DELAY configuration records. The defaults for these will suffice for most operations.

Multiple-System Shared

There are several additions and modifications required for the initialization of multiple shared systems. You may need to add or modify the following records in the MCP system configuration file for shared operations.

PROCESSOR record - This identifies a system's configuration file in stacked configuration decks.

CONNECT record - This identifies which systems will be conected to this system to comprise a multisystem shared system.

DISK record - This declares a disk unit as multisystem shared through a particular SSP.

PACK record - This declares a diskpack unit as multisystem shared through a particular SSP.

UNIT SHARED record - This declares a tape unit as multisystem shared.

SSP DLP record - This declares the SSP to use for multisystem shared configurations.

LIMIT BLT record - This declares the number of entries in the Block Lockout Table.

LIMIT DELAY record - This sets the contention cycle.

These records enable the MCP and other programs to perform concurrent file access and to share peripherals. The system uses the system initialization records when you cold start your system. Refer to the *V Series MCP/VS System Software Operations Guide Volume 1: Installation* for more information about these records.

Shared configuration files may be maintained in two ways:

1. Each system can have its own file. Such is the case when the CONFIG utility is used to maintain the file on floppy disk or the systems are sharing diskpacks only.
2. One system can input and build the configuration files for all shared systems. This is done by a "stacked" configuration deck, where the configuration decks follow one another and are identified by the PROCESSOR record.

Consider the following examples of a two system shared disk and diskpack configuration. In example file 1, the system is named NSHARED and has the processor number 0. In example file 2, the system is named MSHARED, and has the processor number 2. Finally, the example of a stacked deck covers both file 1 and 2. It would be input when system NSHARED is initialized and generate the MCP configuration files (on disk) for NSHARED (where the file is named CONFG0) and MSHARED (where the file is named CONFG2).

File 1

This example MCP configuration file is set up for a system named NSHARED, with the hardwired processor number 0.

```
1 ...HOSTNAME NSHARED
2 ...CONNECT 2
3 ...DLP 34 SSP SSPAAA SSP302 MAILBOX
4 ...DLP O4 DSK HSTLQG
5 ...DISK 4/0 ID 01 SUBSYSTEM 1 DEFAULT SHARED VIA SSPAAA
6 ...DISK 4/1 ID 02 SUBSYSTEM 1 DEFAULT SHARED VIA SSPAAA
7 ...PACK 4/2 ID 03 SHARED VIA SSPAAA
8 ...PACK 4/3 ID 04 SHARED VIA SSPAAA
9 ...DLP 06 MPE
10 ...UNIT 6/1 MPE SHARED.
11 ...UNIT 6/2 MPE SHARED.
12 ...UNIT 6/3 MPE SHARED.
13 ...UNIT 6/4 MPE SHARED.
14 ...DLP 26 MPE
15 ...UNIT 26/0 MPE
16 ...DLP 13 PRN PRN256
17 ...UNIT 13/0 PRN256 PRN
18 ...DLP 01 CRD
19 ...UNIT 1/0 CRD
20 ...DLP 10 CONSOLE
21 ...UNIT 10/4 SPOB ODT LEVEL 9 AD SK 1 HDR MSG D 4
22 ...DLP 12 UNILINE.
23 ...UNIT 12/0 SPOC OCS LEVEL 9 AD SK 1 HDR (WM AM) (MTP DSK DPK PRN)
24 ...USE SLOG AUTO 10000 WRAP
25 ...USE RLOG
26 ...USE BOJ
27 ...USE EOJ
28 ...USE PBD
29 ...USE DUMP DISK
30 ...LIMIT DELAY 30
31 ...STOP
```

File 2

This MCP configuration file is for a system named MSHARED, with the hardwired processor number 2.

```
33 ...HOSTNAME MSHARED
34 ...CONNECT 0
35 ...DLP 34 SSP SSPAAA SSP302 MAILBOX
36 ...DLP O4 DSK HSTLQG
37 ...DISK 4/0 ID 01 SUBSYSTEM 1 DEFAULT SHARED VIA SSPAAA
38 ...DISK 4/1 ID 02 SUBSYSTEM 1 DEFAULT SHARED VIA SSPAAA
39 ...PACK 4/2 ID 03 SHARED VIA SSPAAA
40 ...PACK 4/3 ID O4 SHARED VIA SSPAAA
41 ...DLP 06 MPE
42 ...UNIT 6/1 MPE SHARED.
43 ...UNIT 6/2 MPE SHARED.
44 ...UNIT 6/3 MPE SHARED.
45 ...UNIT 6/4 MPE SHARED.
46 ...DLP 01 CRD
47 ...UNIT 1/0 CRD
48 ...DLP 13 PRN PRN256
49 ...UNIT 13/0 PRN256 PRN
50 ...DLP 10 CONSOLE
51 ...UNIT 10/4 SPOB ODT LEVEL 9 AD SK 1 HDR MSG D 5
52 ...DLP 12 UNILINE.
53 ...UNIT 12/0 SPOC OCS LEVEL 9 AD SK 1 HDR (WM AM) (MTP DSK DPK PRN)
54 ...USE SLOG AUTO 10000 WRAP
55 ...USE RLOG
56 ...USE BOJ
57 ...USE EOJ
58 ...USE PBD
59 ...USE DUMP DISK
60 ...LIMIT DELAY 30
61 ...STOP
```

Stacked Deck

This MCP Configuration file would be input to system N when it is coldstarted. That system would then build the configuration files for both systems, NSHARED and MSHARED.

```
1 ...HOSTNAME NSHARED
2 ...CONNECT 2
3 ...DLP 34 SSP SSPAAA SSP302 MAILBOX
4 ...DLP O4 DSK HSTLQG
5 ...DISK 4/0 ID 01 SUBSYSTEM 1 DEFAULT SHARED VIA SSPAAA
6 ...DISK 4/1 ID 02 SUBSYSTEM 1 DEFAULT SHARED VIA SSPAAA
7 ...PACK 4/2 ID 03 SHARED VIA SSPAAA
8 ...PACK 4/3 ID 04 SHARED VIA SSPAAA
9 ...DLP 06 MPE
10 ...UNIT 6/1 MPE SHARED.
11 ...UNIT 6/2 MPE SHARED.
12 ...UNIT 6/3 MPE SHARED.
13 ...UNIT 6/4 MPE SHARED.
14 ...DLP 26 MPE
15 ...UNIT 26/0 MPE
16 ...DLP 13 PRN PRN256
17 ...UNIT 13/0 PRN256 PRN
18 ...DLP 01 CRD
19 ...UNIT 1/0 CRD
20 ...DLP 10 CONSOLE
21 ...UNIT 10/4 SPOB ODT LEVEL 9 AD SK 1 HDR MSG D 4
22 ...DLP 12 UNILINE.
23 ...UNIT 12/0 SPOC OCS LEVEL 9 AD SK 1 HDR (WM AM) (MTP DSK DPK PRN)
24 ...USE SLOG AUTO 10000 WRAP
25 ...USE RLOG
26 ...USE BOJ
27 ...USE EOJ
28 ...USE PBD
29 ...USE DUMP DISK
30 ...LIMIT DELAY 30
31 ...PROCESSOR 2
32 ...HOSTNAME MSHARED
33 ...CONNECT 0
34 ...DLP 34 SSP SSPAAA SSP302 MAILBOX
35 ...DLP O4 DSK HSTLQG
36 ...DISK 4/0 ID 01 SUBSYSTEM 1 DEFAULT SHARED VIA SSPAAA
37 ...DISK 4/1 ID 02 SUBSYSTEM 1 DEFAULT SHARED VIA SSPAAA
38 ...PACK 4/2 ID 03 SHARED VIA SSPAAA
39 ...PACK 4/3 ID O4 SHARED VIA SSPAAA
40 ...DLP 06 MPE
41 ...UNIT 6/1 MPE SHARED.
42 ...UNIT 6/2 MPE SHARED.
43 ...UNIT 6/3 MPE SHARED.
44 ...UNIT 6/4 MPE SHARED.
45 ...DLP 01 CRD
46 ...UNIT 1/0 CRD
47 ...DLP 13 PRN PRN256
```

```
48 ...UNIT 13/0 PRN256 PRN
49 ...DLP 10 CONSOLE
50 ...UNIT 10/4 SPOB ODT LEVEL 9 AD SK 1 HDR MSG D 5
51 ...DLP 12 UNILINE.
52 ...UNIT 12/0 SPOC OCS LEVEL 9 AD SK 1 HDR (WM AM) (MTP DSK DPK PRN)
53 ...USE SLOG AUTO 10000 WRAP
54 ...USE RLOG
55 ...USE BOJ
56 ...USE EOJ
57 ...USE PBD
58 ...USE DUMP DISK
59 ...LIMIT DELAY 30
60 ...STOP
```

SECTION 17
SORT INTRINSIC PROGRAM (SORT.)

The SORT. intrinsic program provides you with a way to execute rapid disk or diskpack sorts by calling the SORT. program into memory from any program written in ANSI-68 COBOL or in BPL.

When you invoke a sort, the MCP initiates a SORT task to perform the sort and suspends the calling program until the sort completes. For information about the memory used by SORT, see Memory Requirements later in this section.

You can also substitute a user-provided SORT program. The operating system searches the disk directory before calling the bound SORT. intrinsic. If any file or program with the file identifier "SORT." is found, the operating system ignores the bound version and executes the program on disk.

SORT. is designed so that you can modify the SORT intrinsic parameters to match the specific requirements of your site. You can use a file with the reserved name of SORT.. or SORT.n (where n is the system number) for this purpose. If the operating system finds either of these files when SORT. is executed, the parameters in the file are used.

Default SORT. parameters direct work files to 100-byte disks, where they are assigned on a space available basis.

Input

The input file must be wholly contained on a single hardware type. You may not mix hardware types within the same file. For example, a tape file cannot be contained on two different types of tape (MPE, GCR and so forth). SORT. accepts input from any of the following media:

- Cards (coded in BCL or EBCDIC)
- Magnetic tape
- 100-byte disk
- 180-byte diskpack

The following limits apply to SORT. input:

- Maximum blocking factor is 999 records for each block
- Maximum record length is 4,998 bytes
- Maximum block size (record size times blocking factor) is 4,998 bytes

When input records are larger than output records (from SORT.), they are truncated on the right to match the smaller size.

For input files on any media other than disk or diskpack, the number of records to be sorted must be passed from the calling program to the SORT. intrinsic. The SORT. intrinsic assumes this value is accurate. For disk or diskpack input files, the number of records is obtained from the disk (or diskpack) header information.

Additional information required by the SORT. intrinsic includes:

- Disposition of input and output files (type of close)

- Instructions regarding disposition of unreadable blocks in the input data file
- Choice of 100-byte disk or 180-byte diskpack for work areas.
- Instructions regarding sort keys
- Instruction to execute sort with translate if translate is required (see MAKTRN)

Sorting

The SORT. intrinsic arranges records of the input file according to the keys you specify within each record, in the stated ascending or descending collating sequence. Signed numeric keys are sorted algebraically. Negative numbers are less than any positive number.

Sort Key Restrictions

- Sort keys can be mixed, ascending or descending
- Maximum number of sort keys is 40 (subject to any restrictions imposed by the language of the program that calls the SORT. intrinsic).
- Total length of all the sort keys cannot exceed 290 bytes or 580 4-bit "units".
  - Alphanumberic Keys: Each byte in an alphanumeric key counts as one byte or two "units". If the key is signed alpha (represented in COBOL by PIC S9 without COMPUTATIONAL), the high-order (or leftmost) "unit" of the key contains the sign.
  - Unsigned Numeric Keys: Every four bits of an UN field counts as one "unit".
  - Signed Numeric Keys: Every four bits of an SN field counts as one "unit". The sign counts as an additional "unit".

Output

The possible output media from the sorting operation include:

- Cards (coded in BCL or EBCDIC)
- Magnetic tape
- 100-byte disk
- 180-byte diskpack
- Line Printer

Any records you output to a line printer that are smaller than a printer line (132 characters) appear left justified with space fill on the right. Any added content of other output records that you define to be larger than the input records is comprised of all nulls. Output records that you specify as shorter than input records are truncated from the right.

Whenever a group of contiguous input records have equal key values, the order of those same records in the output file is unspecified.

Memory Requirements

Memory requirements for the SORT. intrinsic are as follows:

- Minimum limit is 15K bytes

- Maximum limit is 150K bytes
- Within these minimum and maximum limits, the operating system examines:
  - the memory occupied by the program that called SORT.
  - any value entered with the MEMORY command when the program is executed.

  The operating system allocates SORT. the larger of these two values, as long as the amount of memory does not violate the minimum or maximum limits.

The minimum memory limit is larger if files have record or block sizes that approach the maximum limits for record or block size.

To determine the memory requirement of an object program that uses the SORT. intrinsic, use the information contained in the following File Information Block (FIB) fields as a guide.

- FIBMRL - Record length (in digits)
- FIBMBS - Block size (in digits)
- FIBVAR - Most common record size for variable length record; defaults to FIBMRL (maximum record length) if you do not specify variable-length records.
- MM - Contains 100 if FIBMRL is greater than 1000; otherwise it contains 20.
- SIZE - For fixed-length records, this field contains the value of FIBMRL rounded to the next highest multiple of MM if FIBMRL is not already a multiple. For variable length records, it contains the value of FIBVAR rounded to the next highest multiple of MM, if not already a multiple.
- TOTAL KEY - Size of all keys, in digits.

Using the values from these FIB fields, the minimum memory for the SORT. intrinsic is:

Fixed-length records:

((2 x FIBMBS) + (10 x SIZE) + (10 x TOTAL KEY) + 7000 digits) or, 30,000 digits, whichever is larger.

Variable length records:

((2 x FIBMBS) + (10 x SIZE) + (10 x TOTAL KEY) + 10,000 digits) or, 30,000 digits, whichever is larger.

Disk Requirements

The SORT. intrinsic uses 100-byte disk for scratch work files, unless you specify input parametric information for diskpack (180-byte). The scratch work files used for sorting are about two-and-one quarter to two-and-one-half times the size of the input file.

Sorting, using diskpacks for work files, is designed primarily for multiple diskpack drives to reduce arm movement. You can expect sorting that is limited to a single diskpack drive to have degraded performance.

Special Provisions For Sorting Variable-Length Records

The SORT. intrinsic can accept magnetic tape variable length records as input and can produce them for output. When you use variable-length records, the record size is stored, right justified, in the first four

bytes of each record. The maximum record and block sizes are specified in the FIB.

The MCP blocks the records into the output buffer when it writes a variable-length file, until the buffer cannot hold another record. At that time, the physical write is performed and blocking resumes at the beginning of the buffer.

When the FIB for variable-length file is passed to the SORT. intrinsic, each input record must be treated as being the maximum length (FIBMRL), because variable-length records are not permitted in random disk files. The work files of the SORT intrinsic are created sequentially and accessed randomly.

A provision has been made for optimizing the way the SORT. intrinsic allocates disk for variable-length records. In addition to the maximum record length, the SORT. intrinsic needs to know the most common (that is, most often occurring) record length. This parameter is passed using the FIB through the four digits at FIB+106, known as FIBVAR.

FIBVAR must contain the length, in digits, for the most common record size in the file being sorted. FIBVAR must not be less than the size of the shortest record in the file and cannot exceed FIBMRL (maximum input record length). If you are unsure of the correct value for FIBVAR, you may use the value in FIBMRL. However, when you do this, it disables the disk allocation optimization logic in the SORT. intrinsic.

You should not specify a sort key location that is outside the bounds of the shortest variable-length record you are going to sort.

If the SORT. intrinsic determines that any record input to it is not long enough to contain all the sort key fields specified in the parameters, that record is deleted during the input (distribution) phase. A count of all such deleted records is displayed on the ODT before the merge phase is entered. You will be required to respond to direct the SORT. intrinsic to continue with the truncated input file, or you may elect to DS the program.

Site-specific Sort Parameters

The SORT. intrinsic lets you modify these default parameters by programming them through the SORT BCT as follows:

- Work files can be directed to multiple diskpacks
- Work files can be restricted to a single diskpack
- Area assignments can be specified by FILE, by AREA, by ID number

You can use the utility program DMPALL to modify some of these default parameters to match your own particular site requirements. The modified defaults you establish override the SORT. defaults to create new site-wide SORT. defaults in a default specification file. You create these new site wide defaults through a file called SORT.. or SORT.n, (where n is the system number 0, 1, 2, or 3 to which the file applies).

These files consist of unblocked 80-character records. If either SORT.. or SORT.n is on disk at the time a user program calls the SORT. intrinsic, the file is used by SORT. to modify the standard default parameter set. If both files are on disk, SORT.n is used.

Sample DMPALL card decks are listed immediately below:

```
?PFM CRDDSK PARAMS (SORT..)
?DATA PARAMS
$FILE
$WORK PACK
$PACK LESPAK
?END
```

```
?PFM CRDDSK PARAMS (SORT.2)
?DATA PARAMS.
$FILE
$WORK PACK
$PACK SYSTEM
?END
```

This DMPALL example sets site-wide parameters directing all sort work files to a diskpack named LESPAK. It then establishes parameters for system number 2 that direct work files to a diskpack named SYSTEM.

The available dollar sign cards are $FILE, $AREA, $E.U., $WORK, and $PACK. Although not all cards need to be present in one SORT../SORT.p file, one card of each type can be present. If you request conflicting defaults, only the one that appears last is used. If errors are found in the SORT../SORT.p file at SORT. execution time, a warning message is displayed on the system ODT, and the entire SORT../SORT.p file is ignored. The default SORT. parameters are used.

Listed below are descriptions for each dollar sign card:

$FILE

Directs SORT. to place the entire file on one disk-ID. The disk-ID is selected using a process based on the internal file number (FIBDFN in the File Information Block).

$AREA

Directs SORT. to place each area of the file on a specific disk-ID. The disk-ID is selected using a process based on the area number.

$E.U.

Disk assignment is by disk-ID number. (The designation "EU" refers to the name of an entity analogous to the current disk-ID.) You can enter one or two disk-ID numbers after column 5. Disk-IDs are numbered from 1 to 89. Each number must be two digits long, for example, 01.

- When one number appears, all sort work files are forced to the disk represented by the disk-ID number.
- When two numbers appear, work files alternate between both disk-IDs.

$WORK PACK

Work files are assigned to diskpack. The reserved word PACK must appear after card column 5. User program parameters that you enter through the SORT BCT override this site default option.

$PACK

This card lets you specify multi-file-pack-ids. This card must follow the WORK card. The first multi file-pack-id begins after column 5 on the card, and you can specify up to two of them. The diskpack-id is from one to six characters long. If you use equal signs (=) in the name, they will be considered as masking characters, allowing any one of a number of diskpacks to be used. For example, if you enter a diskpack-id LES===, then all diskpacks with diskpack-ids beginning with LES are available to the SORT intrinsic.

Details on Site-specific Parameters

Spaces function as word delimiters on all dollar cards.

You can disable the default override mode of SORT. at any time simply by removing the file SORT../SORT.p or by changing it to a different name.

The maximum allowable buffer or single block size in any SORT. intrinsic file is 4998 bytes (characters). If you attempt to exceed this limit, it will not be checked by the SORT. intrinsic and may result in an incorrect output.

The SORT work file Wnn000 is a very small file that contains pointers used by the SORT. intrinsic. If you use the SORT../SORT.p file to place SORT. work files on diskpack or a specific disk-ID, the Wnn000 is still assigned to disk.

Error Messages

ASCD/DSND ERROR SORT KEY<text>

A sort key has been specified as other than ascending or descending. The <text>indicates the relative key number and the error information. SORT. intrinsic aborts.

CLASS ERROR SORT KEY<text>

A sort key contains an invalid class. The <text>indicates the relative key number and the error information. SORT. intrinsic aborts.

FIBVAR SPEC ERROR; ALL RECDS SORTED AS MAXIMUM LTH.

Input is variable length and the 4-digit parameter at FIB+106(FIBVAR) does not contain a valid size. SORT. intrinsic continues.

FILE <F-N> IS NOT A TRANSLATE FILE

A check of the header before a translate table file is loaded reveals that the supplied file cannot be a valid translate file. SORT. intrinsic aborts.

ILLEGAL CONTENTS - RECORDS DISCARDED

A key contains the value of "][" (hexadecimal 5A4A). SORT. intrinsic continues.

INSUFFICIENT MEMORY FOR SORT

The SORT. intrinsic has tried to perform the requested job in the memory available and found that it cannot proceed. SORT. intrinsic aborts.

INSUFFICIENT MEMORY SORT OUTPUT

The output block length is too large to be processed in the amount of memory available. SORT. intrinsic aborts.

<integer> VAR RCDS SHORTER THAN SORT KEY DISCARDED: ENTER YES TO CONTINUE SORTING: ELSE DS OR DP

During variable-length sorting, records were encountered that were shorter than the sort key and were discarded. SORT. intrinsic waits for operator action.

Text deleted by PCN-002

INVALID CLOSE CODE TO SORT

The CLOSE type requested for the input or output file is invalid. SORT. intrinsic aborts.

INVALID INPUT FILE USERBLOCK DATA

User-blocking data has been specified for an input file, and this data conflicts with information in the File Information Block (FIB). SORT. intrinsic aborts.

INVALID SIZE SORT KEY<text>

The size of a sort key is invalid. The <text>indicates the relative key number and the error information. SORT. intrinsic aborts.

INVALID LOCN SORT KEY<text>

A sort key has been specified beyond the end of the record. The <text>contains the relative key number and error information. SORT. intrinsic aborts.

INVALID SORT INPUT HDW TYPE

Input must be from cards, tape, 100-byte disk, or diskpack. SORT. intrinsic aborts.

INVALID SORT OUTPUT HDW TYPE

Output must be to cards, tape, 100-byte disk, diskpack or printer. SORT. intrinsic aborts.

INVALID OUTPUT FILE USERBLOCK DATA

User-blocking data has been specified for an output file, and this data conflicts with information in the File Information Block (FIB). SORT. intrinsic aborts.

KEY FIELDS DELIMITER MISSING

The SORT. intrinsic is unable to identify the "last" sort key parameter. SORT. intrinsic aborts.

PARITY ERROR BLOCK <integer>; BLOCK USED
PARITY ERROR BLOCK <integer>; BLOCK DROPPED

One of these messages occurs if a parity error block is used or dropped during SORT. input as a result of a USE or PURGE on ERROR statement in the calling program. <integer>denotes the block number. SORT. intrinsic continues.

PARITY USE OPTION INVALID

An invalid option has been passed for handling parity errors. SORT. intrinsic aborts.

SORT INPUT VOLUME NOT NUMERIC

The number-of-records parameter contains hexadecimal values A-F. SORT. intrinsic aborts.

SORT INPUT VOLUME NOT STATED

The number-of-records parameter is zero, and SORT. input is other than 100-byte disk or diskpack. SORT. intrinsic aborts.

SORT KEY <integer>OUTSIDE RECD;<text>

A key has been specified beyond the record size. The relative key number (<integer>) and information (<text>) are displayed. SORT. intrinsic aborts.

SORT RECORD COUNT ERROR:<integer-1> <integer-2>

The correct record count (<integer-1>) and the error count (<integer-2>) are displayed to indicate that the SORT. intrinsic realizes there is an error. If the same job repeatedly gives the same error count, there may be a hardware malfunction. A Unisys System Representative should be notified. SORT. intrinsic aborts.

SORT TRANSLATE FILE <F-N>IS NOT ON DISK

A SORT using the translate option has been specified, but there is no file on 100-byte disk with the specified file name. Whenever the file is loaded, the sort will proceed. SORT. intrinsic waits for operator action.

VARIABLE RECD SORT INTERNAL ERROR

The SORT intrinsic recognizes that it has made an error. If the same job repeatedly gives the same error message, a Unisys System Representative should be notified. SORT. intrinsic aborts.

VARIABLE RECORD FIB DATA SYNTAX ERROR

This message is displayed and the SORT is aborted if variable-length input is specified and if both input and output are not on magnetic tape, or if the user-blocking bit is set (on). SORT. intrinsic aborts.

ZERO RECORD INPUT TO SORT INTRINSIC

Warning message only. SORT. intrinsic continues.

SECTION 18
TAPE/DISK SORT INTRINSIC PROGRAM (SORT:)

You can use the TAPE/DISK SORT: intrinsic to sort one or more unordered files into a specified sequence. You may specify tape or disk or diskpack media as a sorting medium. The intrinsic is bound to the operating system, and you may invoke it by using the SORT verb from within the program. SORT: is available for use only with the SRTUTL sort utility and with the COBOL ANSI-74 programming language. You can obtain further references about SORT: from the *V Series COBOL ANSI-74 Compiler Programming Reference Manual.*

Normal SORT: operating parameters include work files on PE or GCR magnetic tape, 100-byte disk, or on 180-byte diskpack. Work files on disk are assigned to default subsystems. Work files on diskpack are assigned on a space available basis to system resource (non-restricted) diskpacks.

MEMORY REQUIREMENTS

The memory limits of SORT: are as follows:

- Minimum limit: 25K bytes
- Maximum limit: 499.5K bytes
- Within the minimum and maximum limits, the operating sytem examines:
  - the memory occupied by the program that called SORT:
  - the memory value passed to the operating system in the parameters of the Program Call BCT. (Some compilers make this value default to the size of the calling program.)
  - any memory value entered through the MEMORY command when the program is executed.

  The operating system allocates memory to SORT: based on the largest of these values. If the largest value is outside of the minimum or maximum limit, the operating system uses the limit value.

Pay particular attention to memory used for sorting when large record sizes are involved. Do not minimize sort memory too much when sorting large records. The number of tape or disk I/O operations and the number of sort passes depend on the number of records that can be evaluated in memory during the sort. As record sizes approach the maximum design value for a given program, the number of records that can be evaluated becomes relatively small if the memory available to SORT: is too minimal. For example, if the record size is 4000 bytes or more, an invocation of SORT: with less than 100K bytes of memory will result in less than optimal sort performance.

KEY SPECIFICATIONS

SORT: arranges records of the input files according to specified keys within each record in the ascending or descending collating sequence that you specify.

The sort key may consist of up to 40 separate fields, according to the following rules:

The size of the combined keys may not exceed 290 bytes allocated as follows: One byte for each two digits of any computational key item. One byte for each two digits of any signed item plus one digit for the sign.

(Both of the above are rounded up to the next whole byte.) One byte for each character of any other class of key item. Special COBOL trailing sign bytes and separate leading sign bytes are permitted. For these key types, the sign counts as a whole byte.

No single key field may exceed 98 units in length. In this context, units mean digits for computational items and bytes for all others.

No sort key location should be specified that is outside the bounds of the records to be sorted. For variable-length records, this means the shortest record to be encountered throughout the run. If while reading the input file, the SORT: intrinsic determines that any input record is not long enough to contain all of the sort key fields specified by the calling program, that record is deleted during the input phase. A count of such deleted records is displayed after the last input file has been read, and you will have the opportunity to continue the sort without the offending records.

All signed key fields are treated as computational items. If they appear in the record description as signed eight bit, they will be packed into signed four bit prior to key comparison.

All signed keys are sorted algebraically (negative numbers are less than any positive number).

INPUT AND OUTPUT REQUIREMENTS

SORT: can accommodate input and output to and from cards, pseudo deck, tape, disk or diskpack medium supported by the operating system.

INPUT RESTRICTIONS

Any one input file must be wholly contained on one physical media type. Keeping this restriction in mind, you may intermix any media and physical block sizes in any combination. The record size (FIBMRL) must be identical for all files participating in the sort. For variable length files, the maximum record length for the output file may not be less than that for any of the input files.

The actual files you want to sort must agree with the declarations of those files as specified in the calling program. For example, if you declare an input file with one blocking factor and the actual file presented is found to have some other block size, the SORT: program will error halt.

When you specify multiple input files, you must still specify only one output file. If the calling program requires that more than ten files be input to SORT:, the sorting memory requirement is increased by 500 bytes over whatever memory would otherwise be needed for each file over the tenth.

Tape files that consist of variable-length records that are supported by the operating system may be processed by SORT:. However, if you are processing such variable-length records, you must specify all files that participate in that sort as variable length files. If you use variable-length records, the record size is stored right justified in the first four bytes of each record. The MCP blocks the records into the output buffer when writing a variable length file until the buffer cannot hold another record. When this occurs, the physical I/O is performed, and blocking resumes at the beginning of the buffer.

When the FIB for a variable-length file is passed to SORT:, each input record can be treated as the maximum (FIBMRL) length since variable-length records are not permitted in random disk files. (SORT: work files are created sequentially and accessed randomly.) To optimize the allocation of scratch disk for variable- length records, the SORT: intrinsic requires the calling program to supply the length of the most common (that is, most frequently occurring) record, in addition to the maximum record length.

The most common length is passed to SORT: in one of the interface parameters. When this value is provided, each variable-length record is broken into fragments of that value. These smaller sized fragments are processed throughout the merge passes, and result in a considerable reduction in both disk space required and time needed for the sort.

The maximum block size for SORT: is 19998 bytes. When you use large block size files, it increases the memory required for the program. The maximum record size that can be sorted is in excess of 4000 bytes. The precise record size value depends on whether you are using tape, disk, or diskpack as a scratch medium, according to the following guidelines:

- tape sort - 4998 bytes maximum
- disk sort - 4800 bytes maximum
- diskpack sort - 4860 bytes maximum

INPUT AND OUTPUT ASSUMPTIONS

Card, magnetic tape, disk, and diskpack input files are assumed to be in EBCDIC code. Binary or ASCII input files are not acceptable as input. Blocked files on tape may consist of a variable number of fixed length records (frequently the last block on a tape is a short block). The only exception is variable-length records on tape (where each record carries its own length in a 4 byte container at the front of the record).

RULES FOR SEQUENCE OF OUTPUT FILES

The relationship is unspecified between the records in the output file within a given group of contiguous equal key records and the ordering of those same records in any input file.

Tape Sorting

The TAPE SORT mode of SORT: utilizes from 3 to 8 work tapes in the sorting operation. You may specify the number of tapes in the interface parameters, but you cannot alter them at run time. If an input file is on tape or the output file is to tape, that unit can serve its input function prior to being used as a work tape or its output function after having served as a work tape.

Work file units may be 9 Track NRZ, PE, or GCR type only, and you may inter mix them in any combination. If you specify the sorted output as tape, that output will usually be directed to the tape unit that was first made available as one of the work tapes.

Tape Sort Operation: A short narrative description

First, SORT: acquires and validates the control parameters from the caller. Next, the work file tapes are OPENed. The number of work tapes that are needed is one less than the number specified for the given sort.

Then, the input files are opened and read one at a time, and the records are written onto the work tapes in sequenced runs. When the last input file has been read, the last work file is opened, and a series of merge passes are performed on the work files.

When it is determined that the number of runs remaining is less than the number of work file tapes, the output file is then opened. The final merge then writes the sorted file to the media specified by the user.

Operating Considerations

It is a good idea to have all of the scratch work tapes large enough to hold the entire sort input. This because no work file can be a multireel file. If sort input is originally very close to being in sequence, most of the sort data may end up on one single work tape since work file output reel swap during the input phase occurs only when a new sequenced run is to begin.

If sort data is more or less random, the total number of records that can be sorted in a single sort can be increased by increasing the number of tape units available to the program.

DISK OR DISKPACK SORTING

Disk Sort Media

The DISK/PACK mode of SORT: utilizes work files on:

- 180-byte diskpack
- 100-byte HPT disk or LAK diskpack

Default Media

The default sort media is 180-byte unrestricted diskpack. If the calling program so specifies, 100-byte default disk will be used instead. The work file media will always be on one type of media only, either all 100 or all 180-byte.

Work File Requirements

The amount of work file space required for a disk sort is approximately 2.5 times the size of the combined input files. If the work files you are using are on removable media, all volumes of the media must be present during the entire sorting operation.

Disk Sort Operation: A short narrative description

The SORT: program acquires and then validates the control parameters from the caller. Next, memory available is apportioned according to the number of input records that are expected. Once this is determined, a multi area disk or diskpack work file is opened large enough to hold the expected number of records.

Then, the input files are opened and read one at a time, and sequenced runs written to the work file. When the last input file has been read, another allocation of resources occurs. The resource allocation is based on the number of records actually read and the number of merge passes that will be needed to merge the sequenced runs into a single run.

The intent of this allocation is to minimize the number of merge passes consistent with the memory available to the program. Each merge pass reads a work file, which produces fewer runs of longer length, and then writes an output work file. These merge passes continue until it is observed that the next pass is to be the last. When the final merge is reached, the output file is opened, and the sorted file produced.

Operating Considerations

SORT: does not use a specific method of work file allocation unless default media override is being used. (Default media override is discussed in the next section below.) If the work files end up on the same spindle or on disk units already experiencing a high level of activity, the performance of SORT: will be adversely affected. If work files are assigned to a disk-ID, which is also the prime disk-ID containing the disk directory, then overall system performance will be adversely affected.

Default Media Override

There is no facility in the SORT/MERGE interface that lets you control which specific disk/diskpack storage units should be used for sort work files. One reason for this is that such unit specific information is frequently unknown when a program is compiled. Also the amount of space available for work files tends to vary significantly from spindle to spindle over time. For these reasons, assigning work files to specific media is deferred until run time.

Because of this assignment delay, and because the MCP makes assignment decisions on a "space-available" basis, the media where the work files get assigned may not always be optimal. This can cause particular problems when the spindles with the most space available are the ones being accessed most frequently.

Due to these potential problems, a media default override feature permits all decisions you make about sort intrinsic work files to be based on parameters that you supply to the system at execution time. Keep in mind, though, that this feature is similar, but not identical, to the one implemented for the earlier SORT. program. The present SORT: program utilizes a control file that affects only its operation. The syntax of this control file and the control file for SORT. are also similar, but not interchangeable.

The media default system is activated by the presence of a small control file on the system disk. The name of this file is in the form of"SORT:x." The first five characters are always the same. The last character ("x" in this example) can be a digit 0, 1,2, or 3, in which case the named control file will apply only to the system represented by that system number. If the last character is a colon, the control file will apply to all systems. When both SORT:x and SORT:: are present, SORT:x takes precedence.

The control file is created via the DMPALL utility as follows:

```
? PFM CRDDSK PARAMS (SORT:x) where x = system number

? DATA PARAMS

one or more parameter records.
? END.
```

The various parameter records, their syntax, and a brief description of their function are presented below. Not all types need be present. The sequence of the various records is not significant.

$FILE

Directs SORT: to place the entire file on one disk-ID (sets FIBDTK = 1 in the File Information Block). The disk-ID is selected using a process based on the internal file number (FIBDFN in the File Information Block). You cannot specify $FILE together with $AREA or $E.U.

$AREA

Directs SORT: to place each area of the file on a specific disk-ID (sets FIBDTK = 2 in the File Information Block). The disk-ID is selected using a process based on the area number. You cannot specify $AREA together with $FILE or $E.U.

$E.U.

Disk assignment is by disk-ID number. (The designation "EU" refers to a previous entity analogous to the current disk-ID.) You can enter one or two disk-ID numbers after column 5. Disk-IDs are numbered from 1 to 89. Each number must be two digits long, for example, 01.

- When one number appears, all sort work files are forced to the disk represented by the disk-ID number.
- When two numbers appear, work files alternate between both disk-IDs.

$PACK <family1> <family2>

Assigns diskpack work files to the family names 1 & 2. These may be restricted diskpacks. If you specify only one name, all diskpack work files will be directed to that diskpack family.

- $WRKP

  Ignores the work file type as specified in the calling program. The setting of the system option WRKP will determine a work file assignment. Work files are assigned to 100-byte disk if WRKP is reset, or to 180-byte diskpack if WRKP is set. You cannot specify $WRKP together with $WORK. (Refer to the *V Series MCP/VS System Software Operations Guide, Volume 1: Installation* for a description of the USE WRKP system parameter.

$WORK DISK
$WORK PACK.

Ignores the work file type as specified in the calling program. The media selection word DISK or PACK determines whether work files are assigned to 100-byte disk or 180-byte diskpack. $WORK cannot be specified together with $WRKP.

$SUPR

Suppresses the control file error messages after the first sort to sense the error.

The command word occurs in columns 1-5 of each record. The remainder of the record is free-format. Spaces or commas are delimiters. The work media control file will be checked at execution time for errors and inconsistencies. Any error found will cause the file to be ignored. For example, the $FILE and $AREA options cannot both be present nor can they coexist with the $E.U. option.

The $WORK and $WRKP options are mutually exclusive. Such errors, or the inclusion of unidentified records, will result in the following error message: "SORT DEFAULT MEDIA CONTROLFILE ERROR." In addition, if you specify specific disk-ID numbers or diskpack family names, those media must actually be present at sort time. Otherwise, the message "NON-PRESENT DISK OR PACK REQUESTED AS SORT WORK AREA" will be displayed. Any error will cause the entire media override feature to be inoperative. If the warning suppress record $SUPR is present, an indication of problems with the media override feature will be displayed only by the first sort task to encounter the error.

The default media for sorting is 180-byte.

VIRTUAL COLLATING SEQUENCE

With this option you can alter the sequence in which SORT: orders records during the sorting process. Normally, all characters encountered in the sort keys are arranged in the hardware collating sequence of B2900/B3900/B4900/V Series Systems (i.e., 00 through FF). Only those elements of the sort key described as unsigned alphanumeric are affected by the translation capability. Computational sort keys are always processed according to the hardware collating sequence.

The virtual collating sequence lets you:

- Specify an entire new collating sequence for the particular program invoking SORT:.
- Retain the normal collating sequence of 00 through FF, except for certain characters whose rank in the sequence you want to interchange.
- Make a number of characters have the same rank for the ordering of records.

The features of the virtual collating sequence option are particularly useful when you are using foreign alphabets or conversions from other computer systems.

Translate Option

You can invoke the SORT: intrinsic with the Translate option by specifying the name of the translate table file in the SORT CALL parameters. When you do this, SORT: acquires this file during initialization and uses it during all subsequent key processing. In this way, records are ordered as though the hardware collating sequence had been the virtual sequence that you specified.

Translate Tables

Any Translate Tables you want to use with SORT: (prepared by the MAKTRN program) must be on disk at the time you invoke SORT:.

The names of the Translate Table Files should be unique so that the files are not inadvertently sorted into the wrong sequence. The Translate Table File consists of a 400-byte single record, single area file on disk. The SORT: intrinsic verifies the header information prior to OPENing the Translate Table File to ensure that the file is a 400-byte single record, single area file.

ERROR AND WARNING MESSAGES

All failure terminations of SORT: display an error message. For recoverable errors, you will have the option to continue the sort and the task that invoked it in spite of the error. For all other errors, a memory dump is produced, and SORT: proceeds to error termination, thus canceling the caller and whatever job system it pertains to.

Non-Fatal Errors

Below are two non-fatal error messages that are displayed:

INPUT RECDS BEGIN WITH SORT MARKER CHAR

No input record may have the reserved character combination][(5A4A) as its first two characters. This error message indicates that input records were encountered that violate this rule. Refer to RECOVERY below to respond to this error.

RECORDS SHORTER THAN LAST POSITION OF SORT KEY

This error occurs during processing variable length records when it is determined that one or more input records are shorter than some portion of the sort key (example: a sort key begins at byte 102 and a 100-byte record appears in the input file). Refer to RECOVERY below to respond to this error.

RECOVERY

During the input phase, all records that do not follow the requirements presented in this section are discarded. A count is kept of the number of records discarded. The error messages above will be displayed (as appropriate) when all input has been read. You will have the option to continue the sort without the records that produced the errors. If you choose to continue, the sort file will be produced and control returned to the caller. If you choose not to continue, the sort will be aborted.

Fatal Errors - Potentially Correctable

The following fatal errors have been provided based on situations observed and potential problems anticipated during development of this product. These error messages are self explanatory and the errors are potentially correctable by the user. To recover, correct the error and restart the sort.

ALL FILES NOT DECLARED SAME RECD SZ

Except for variable length sorts, all files included in the sort must have the same record size (FIBMRL). This error can also occur if the block size or record size for an input file does not exactly match what was declared for that file in the calling program.

KEY OUTSIDE RECORD OR SIZE ERROR

A sort key has been specified that has either zero length or is too long or all or part of it is beyond the maximum record size FIBMRL.

MUST SPECIFY INPUT RECDS EXPECTED

Input is not from disk or diskpack. Therefore, file size cannot be determined from a header and must be stated explicitly.

NUMBER OF TAPES FIELD INVALID

Number of tapes not within expected parameters of 3-8.

OUTPUT FILE IS DECLARED TOO SMALL

The capacity of the output file as declared is insufficient to hold the number of records expected.

REC/BLK SIZE NOT AS DECLARED IN FIB

For disk and diskpack input files, the header is checked at open time to verify that real record size equals stated record size.

SORT KEY CLASS SPECIFIER INVALID

A key field has been found that is not UN, UA, or any other permitted value.

Fatal Errors - Non-Correctable

The following fatal errors are not normally correctable by the user. The error may be an interface error, possibly caused by the compiler which invoked the sort. The error may also be an internal SORT: error.

If the specific error is inconsistently produced, it may be the fault of a hardware deficiency.

If the specific error is consistently produced, and changes as you modify data, memory size, or sort keys, it is possible that a program deficiency is the cause. In such cases, memory dumps, and copies of the data and calling program should be saved for analysis.

APPARENT MISSING TAPE BLOCK
BCT 0794 FAIL OR PARAM SIZE ERROR

Program call BCT failure. Parameters passed do not match MCP expectations.

DISTR PHASE MEM ALLOCATION ERROR
DUMMY RUN ASSIGNMENT FAILURE
DYNAMIC CODE RELOCATION FAILURE
DYNAMIC FIB RELOCATION FAILURE
INTERNAL SORT ASC/DESC FAILURE
MEMORY (DYNA) ALLOCATION ERROR.
MERGE PHASE MEM ALLOCATION ERROR
MERGE PHASE RUN CONTROL FAILURE
MERGE PHASE SEEK CONTROL ERROR
MERGE PHASE UNBLOCKING FAILURE
NOT ENOUGH MEMORY (PARAM READ)
NOT ENOUGH MEMORY (SORT BUFFERS)
OUTPUT PHASE MEM ALLOCATION ERROR
PASS TO PASS RECORD COUNT ERROR

These errors can result from internal inconsistencies in SORT:. Contact your Unisys Customer Service Engineering representative.

SMDP FIELD INVALID OP SPECIFIER
SMDP OP=01 MISSING OR DUPLICATE
SMDP OP=28,29 MISSING OR SEQ ERR

Sort key specifiers have been encountered where not expected, or there are none at all. This error can occur when a calling program that is compiled with a particular version of COBOL that is now running on a version of the MCP that invokes an incompatible version of SORT:.

SMDP OP=31,35,36 MISSING OR ERR.
TRANSLATE FILE SPEC INV OR MISSING

Translate file name is too long or the bracketing characters required around it in the parameters are missing.

VARIABLE LENGTH PARAMS DO NOT AGREE

A variable length file has been encountered that was not expected. If variable length sort is active, all files participating in that sort must be specified as variable length files. (FIBBLK = 2).

VAR RECD INPUT PROCESS ERROR
VAR RECD MERGE PROCESS ERROR
VAR RECD OUTPUT PROCESS ERROR

Failure detected in variable record management. Portions of records were detected in unexpected locations.

SECTION 19
GENERALIZED SORT UTILITY PROGRAM (SRTUTL)

SRTUTL is designed to SORT indexed sequential, relative, or sequential files by requested keys and to generate either an ADDROUT file acceptable to RPG, or a new sequential file of complete records. Though SRTUTL may be used by both RPG and COBOL users, many of the functions provided by SRTUTL are available within the COBOL compiler.

Indexed and relative files that you want to SORT with SRTUTL must have been created by a program compiled with a compatible release of COBOL or RPG. Additionally, you may selectively choose or drop input records before you actually begin the SORT.

SYNTAX

SRTUTL accepts both a primary and secondary source file defining the SORT requirements and then generates a sorted file in the format you request.

The input to SRTUTL consists of a series of program statements that are shown in Figure 19-1 below.

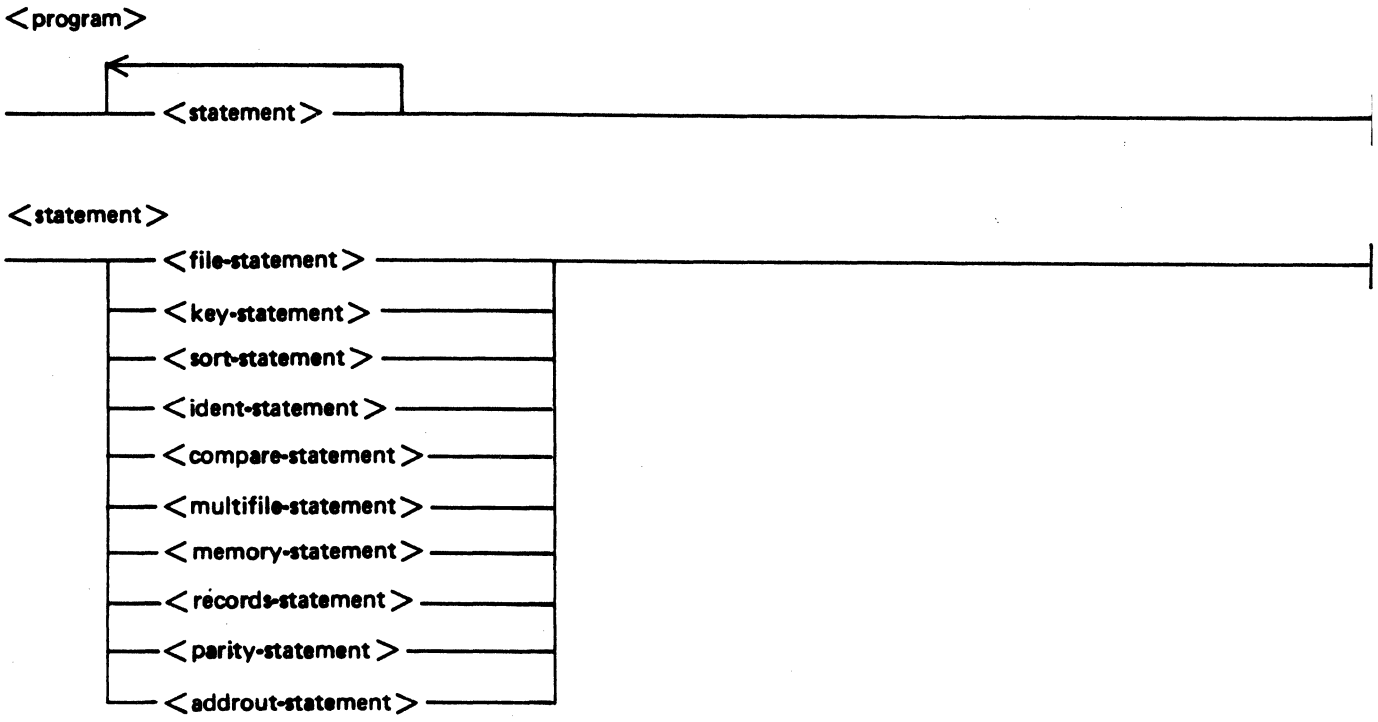


Figure 19-1. SRTUTL Program Statements.

These statements may appear in any order as input to SRTUTL. You enter the program statements in free format into positions 1-72 of an input record. Positions 73-80 are reserved for sequence numbers.

Anything that you enter following a percent sign (%) is treated as a comment, and is ignored by SRTUTL. These comments are printed on the output listing.

There is no EDITOR file type reserved for SRTUTL. If SRTUTL parameters are to be maintained in EDITOR files, file types whose sequence numbers are in positions 73-80 (ADS, BPL, FORTRAN, DASDL, WFL) should be used.

If the DASDL format is used, SRTUTL jobs can be included through SYS COMP since the compile commands are identical.

FILE Statement

The file statement allows you to describe the format both for the input file you want to SORT, as well as for the output file that results from the SORT. You should take care to beas accurate a possible when you enter the descriptions. This is especially true when you are describing indexed and relative files since these files types store information in a control block in the maximum record position of the file.

All FILE statements you enter must include the medium, the record length, and the blocking factor for each file. You must also specify in the statement either the file identifiers or the close with PURGE disposition of the input file, or both.

Additionally, you must begin the FILE statement with the reserved word FILE followed by the reserved word IN and an input file description; this in turn, must be followed by the reserved word OUT, and an output file description.

The syntax of the FILE statement is shown in Figure 19-2 below.

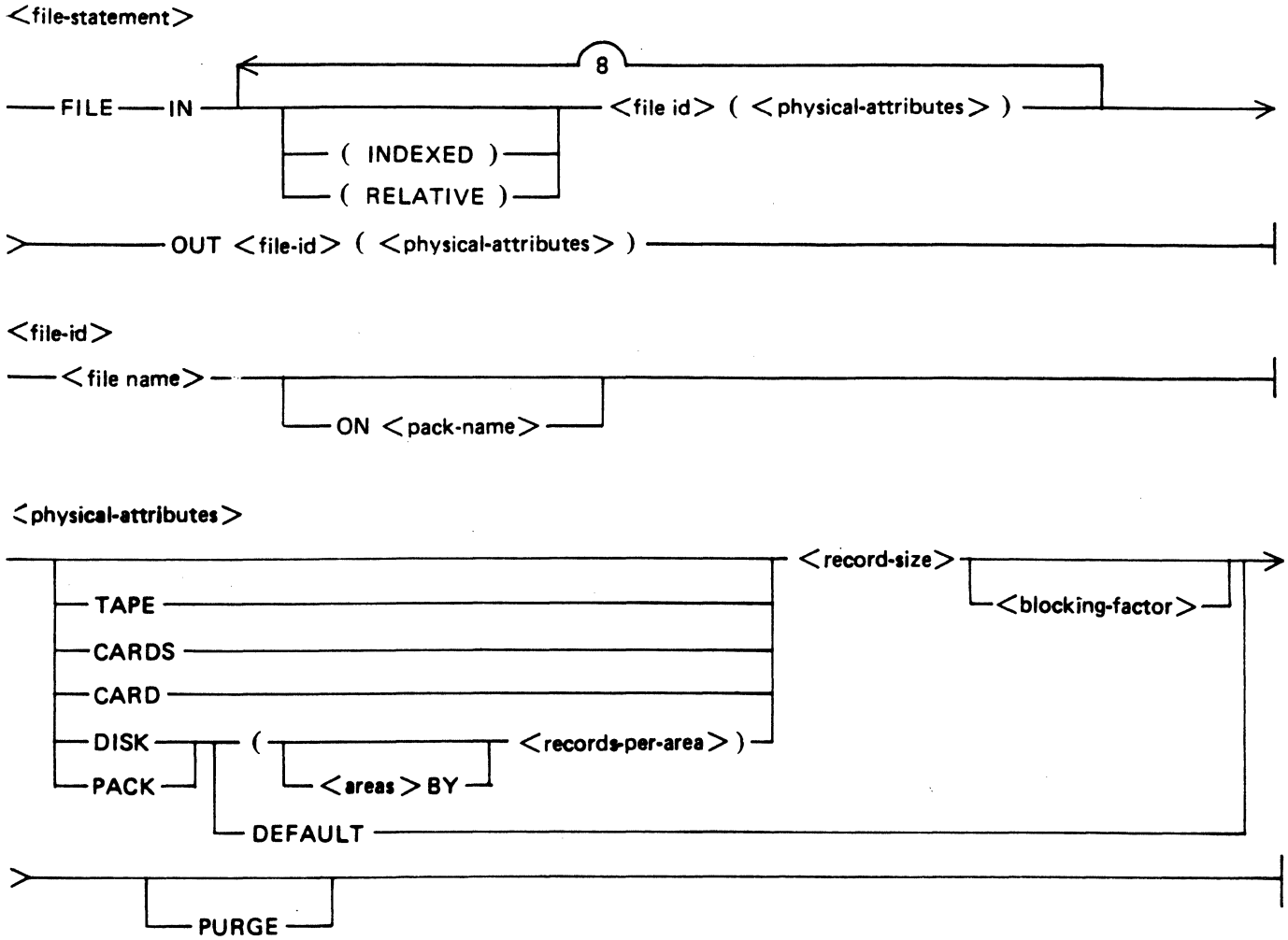


Figure 19-2. FILE Statement

The components of the FILE statement that are depicted above are described in the following paragraphs.

1. FILE TYPE - You may use this parameter to identify the input file as an INDEXED or RELATIVE file. If you omit this parameter, then the file is assumed to be a normal sequential file.

2. FILE ID - You must be sure that this ID label conforms to the standard format of one to six alphanumeric characters, the first of which must be alphabetic. If you enter any special characters, or if the identifier is a reserved word, you must enclose it in quotation marks.

3. MEDIUM - You may use this parameter to define the hardware where the input and output files reside. You must enter one of the following reserved words in the definition:

   DISK - Disk file

   PACK - diskpack file

   TAPE - Tape file

   CARD or CARDS - Card file

4. DEFAULT - You may invoke default settings for both an input and output file.

   Input file: If the input file is on disk or pack, you may specify DEFAULT. The number of areas, records per area, record-size, and blocking factor will be automatically obtained from the disk file header. If this information is not available, you must specify the number of records for each area. The default number for the number of areas is set to 100.

   Output file: The default records per block for the output file is set to 1. The number of areas is set to 20, and the record per area is set to 2000. The individual record length is equal to the record length that you have established for the input file.

5. AREA AND AREASIZE - If you do not use DEFAULT, you must specify the number of records per area for all output files and for any input file on disk or diskpack. Also, you may elect to specify the number of areas; if you do, you must enter the word BY following the number. The default number of areas is 100.

6. RECORD SIZE - If you do not use DEFAULT, you may use this parameter to specify the logical record length in characters. This specification is required. Input records may not be of variable length. The maximum record length accepted by SRTUTL is 4,998 characters. If the record size you enter is not evenly divisible by 2 (that is, an even number of words), the size is padded by SRTUTL.

7. BLOCKING FACTOR - You enter the block size to define the number of logical records in a block. This value may not exceed 999 per block. If you omit a blocking factor, SRTUTL assumes fixed-length records, blocked 1.

8. TYPE OF CLOSE - You have the option to specify the disposition of the input file as close with PURGE. You indicate this by entering the word PURGE. If you omit this entry, the object program will close the input file with RELEASE.

You should enclose in parentheses and delimit by a space all parameters that indicate media, record length, block size, and disposition.

If you declare more than one input file, but do not declare a SORT statement, then the two files will be merged.

Below is an example of a typical FILE statement:

```
FILE IN(INDEXED) "DISK" (DISK(20) 200 10)

OUT SORTED (PACK (20 BY 100) 200 5)
```

This FILE statement describes an indexed sequential input file called DISK. The file resides on disk and has 20 records per area and is comprised of 200-character records blocked 10. The output file is to be written to diskpack (20 areas, 100 records for each area) to be comprised of 200 character record blocked 5. Note that the input file-id "DISK" is enclosed in quotes because DISK is a SRTUTL reserved word.

MULTIFILE Statement

The MULTIFILE statement allows you to name additional input files that you want included in the SORT. You must use the same format in this statement as you used in the FILE statement to define the input file. The MULTIFILE statement is not valid when you also define that a merge be done.

The syntax of the MULTIFILE statement is shown in Figure 19-3 below.

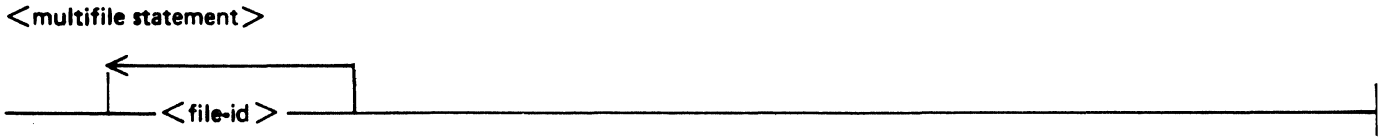


Figure 19-3. MULTIFILE Statement

KEY Statement

The KEY statement lets you describe the keys by which the file should be SORTed. You must begin the key description with one of the reserved words: FIELD, FIELDS, KEY, OR KEYS, followed by a description of each key.

The syntax of the KEY statement is shown in Figure 19-4 below.

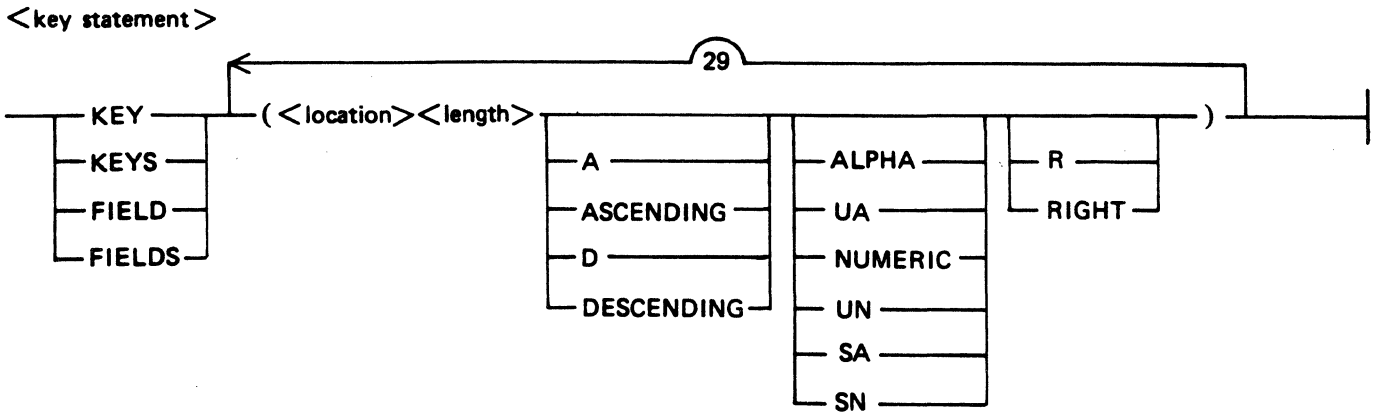


Figure 19-4. KEY Statement

The components of the KEY statement that are depicted above are described in the following paragraphs.

1. LOCATION - The location is the most significant character or digit of the SORT key related to the beginning of the record. The first character or digit (depending on the data type you are using) of the record is considered to be location 1. If the field you want to SORT is signed numeric, the location is the sign digit. For a SORT on assigned alphanumeric data, the location is the character containing the sign.

2. LENGTH - You may use this parameter to specify the number of characters or digits (depending on the DATA TYPE) in the SORT key. You must include this parameter. The LENGTH of a signed numeric field does not include the sign digit, but the sign character is included in the LENGTH of assigned alphanumeric field.

3. SEQUENCE - You may use this parameter to specify the sequence to which the field should be SORTed. An A or ASCENDING is coded for ascending sequence; a D or DESCENDING is coded for a descending sequence. If you omit to enter the code, an ascending sequence is assumed. You may mix ascending and descending sequence SORT keys within the total key.

4. DATA TYPE - You can specify the type of data to be SORTed by entering one of the following reserved words:

   ALPHA OR UA: for unsigned alphanumeric (data is in bytes)

   NUMERIC OR UN: for unsigned numeric data (data is in digits)

   SA: for unsigned alphanumeric data (the sign is in the most significant digit of the most significant character and the data is in bytes)

   SN: for signed numeric data (data is in digits)

5. SIGN POSITION - If you do not enter an R or RIGHT here, the data will be defined as left signed SA or SN. For right signed SA data, the sign will be in the most significant digit of the rightmost character. For right signed SN data, the sign will be in the rightmost digit. It is invalid for you to enter an UN or Alpha data type in this position.

You can specify different DATA TYPEs for individual keys within a total key. If you omit the DATA TYPE, the unsigned alphanumeric is assumed.

You should enclose each field description in parentheses and separate them from each other with a space. You can delimit the parameters within an individual description by a space.

The individual number of SORT keys that you are permitted by SRTUTL is 30. The total key (that is, the field that results from the concatenation of the individual keys) must not exceed 100 units. The units are bytes (characters) for UA and SA keys; whereas, the units are digits for UN and SN keys. The combined length of all keys, including any padding digits necessary to align keys to byte boundaries, must not exceed 560 digits.

Fields that you enter which overlap are not acceptable. For example, the specification KEY (3 5) (4 6) will be flagged as a syntax error.

NOTE

A key containing the character combination "][" (the characters right bracket-left bracket; the EBCDIC representation of hexadecimal 5A4A) is illegal.
Below is a typical example of a KEY statement:

```
KEY (5 5) (25 2 SN) (30 3 D)
```

This KEY statement causes a SORT that proceeds in the following manner: First, by major key, which is five unsigned alphanumeric characters beginning in the fifth character position of each record, into ascending sequence; then it by two signed numeric digits whose sign is located in the 25th digit position of each record, into ascending sequence; and then finally by the minor key, three unsigned alphanumeric characters beginning in the 30th character position of each record, into descending sequence.

SORT Statement

The SORT statement allows you to describe the type of SORT you want performed. If you omit this statement, then the HOTSORT will be assumed, unless you declare more than one input file in the FILE statement. In this case, a merge will be performed.

You may define up to a maximum of eight files as input to merge. It will be assumed that each of the input files you define are already in the desired sequence.

The syntax of the SORT statement is shown in Figure 19-5 below.

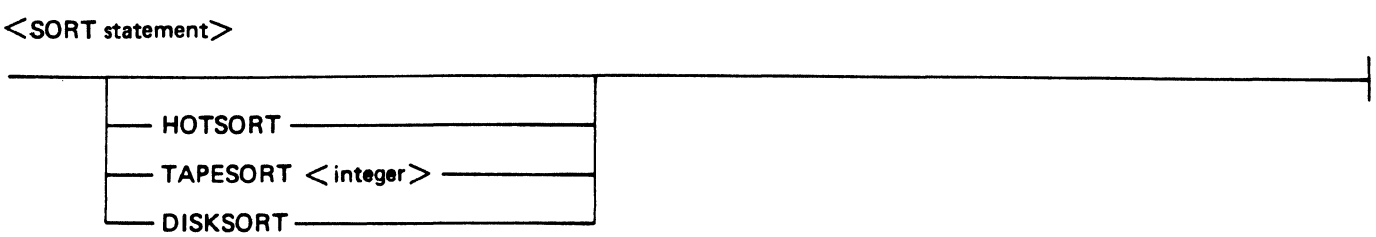


Figure 19-5. SORT Statement

HOTSORT will read a relative, indexed, or sequential data file sequentially, and will call the other sort utility SORT: to sort the records, using disk for the work files.

TAPESORT specifies that <integer> tapes will be used for the SORT. The number of tapes you enter must be between 3 and 8.

DISKSORT will read a relative, indexed, or sequential data file, and will call the system SORT "SORT." to SORT the records, using disk for the work files.

If you omit the INCLUDE or DELETE statement, the SORT will be called directly. Otherwise, an intermediate disk file with the same name as the output is created, and the number of records to be SORTed is displayed.

ADDROUT Statement

You can use the ADDROUT statement, as related to RPG, to automatically select the requested SORT keys (see KEY statement above), concatenate them, and then write a file of records consisting of an 8 digit relative record number, plus the concatenated keys. Next, ADDROUT will call the requested system SORT to SORT on the concatenated keys. The default SORT is HOTSORT.

The syntax of the ADDROUT statement is shown in Figure 19-6 below.

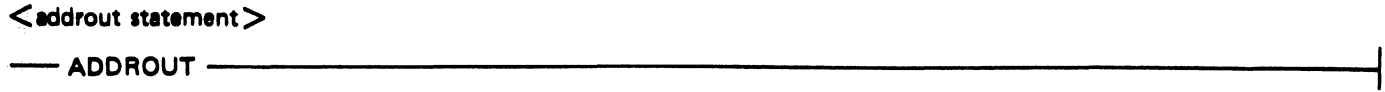


Figure 19-6 . ADDROUT Statement

IDENT Statement

The IDENT statement lets you assign a specific name to the SRTUTL output listing. If you omit this statement, spaces will appear at the top of the listing instead of a program name.

The syntax of the IDENT statement is shown in Figure 19-7 below.

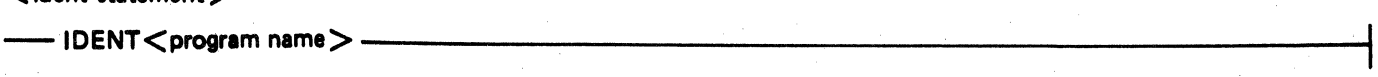


Figure 19-7. INDENT Statement

The program-name you enter must be from one to six alphanumeric characters, the first of which must be alphabetic. If you use special characters in the program name or include a reserved word, you must enclose it in quotation marks.

COMPARE Statement

You can use the COMPARE statement to specify the inclusion or exclusion of records.

The syntax of the COMPARE statement is shown in Figure 19-8 below.

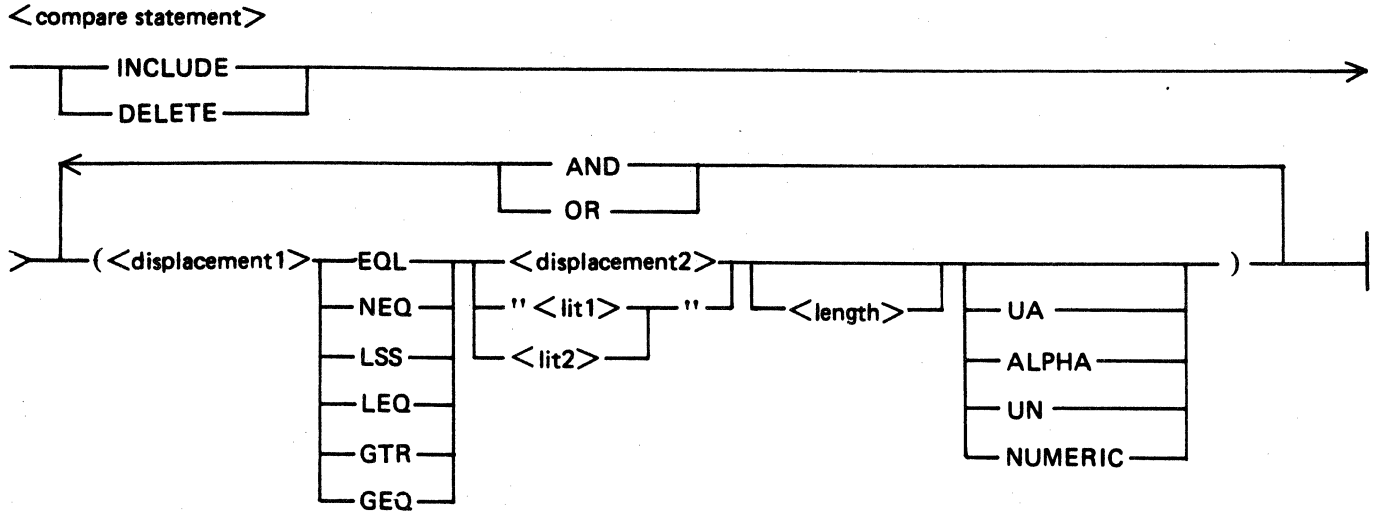


Figure 19-8. COMPARE Statement

The words INCLUDE and DELETE are reserved words, and you may enter them only once within a set of SORT specifications. However, you may follow each of these words with one or more keys. If you specify both INCLUDE and DELETE, the INCLUDE will be applied first.

The total number of keys you specify may not exceed ten. You can join multiple keys within INCLUDE or DELETE by using the reserved words AND or OR. The resulting expressions are then evaluated element by element, left to right.

CAUTION

If an OR is encountered and all previous expression elements are satisfied, the evaluation is terminated as TRUE. If an AND is encountered and all previous expression elements are not satisfied, the evaluation is terminated as FALSE. In all other circumstances, the entire expression is evaluated to a TRUE or FALSE conclusion.

You can use the first operand (displacement 1) to specify the offset of the key from the beginning of the record. The offset will be in bytes if you specify ALPHA as the key entry. If you specify NUMERIC as the key entry, the offset will be in digits.

You can specify the length of the field in the length field. If you omit this specification, the default value is 1. The entry that follows the key displacement is the relational operator that is applied to the comparison of the specified operands. Valid entries are EQL (equal), NEQ (not equal), GTR (greater), GEQ (greater or equal), LSS (less), and LEQ (less or equal).

You can use the second operand (displacement 2) to specify the displacement into the record or as an alphanumeric or hexadecimal literal. As a displacement, the second operand is subject to the rule outlined for the first operand. Hexadecimal literals are delimited by at-signs (@), and may contain one to six hexadecimal digits. Character literals are delimited by quotation marks ("), and may contain up to three characters. All characters are valid with the exception of the quotation or question mark. Literals are padded or truncated as necessary, depending on the setting of the length attribute.

You can use the length field to specify the length of the operands to be compared. If you omit this specification, the default is 1.

You can use ALPHA and NUMERIC to specify whether the operands are character or digit oriented. You can abbreviated these entries to UA and UN, respectively. If you omit this specification, the default is UA (ALPHA).

MEMORY Statement

You can use the MEMORY statement to increase the memory to be used by the SORT intrinsic. The default for this setting is digits. Normally, SORT will use the same amount of memory as SRTUTL. MEMORY can be used to increase the amount to be used by SORT: (up to 300 KB).

The syntax of the MEMORY statement is shown in Figure 19-9 below.

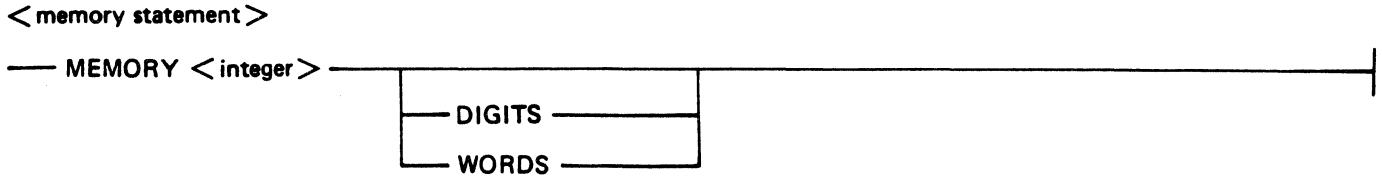


Figure 19-9. MEMORY Statement

PARITY Statement

The PARITY statement lets you drop all unreadable records that are identified during the actual SORT. The default value for this statement is that option is not set. In this case any parity errors will cause the program to be discontinued.

The syntax of the PARITY statement is shown in Figure 19-10 below.

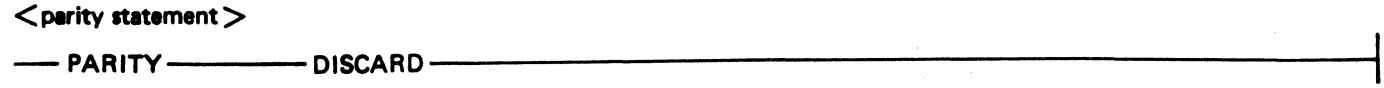


Figure 19-10. PARITY Statement

RECORDS Statement

The RECORDS statement lets you optimize the SORT operation by supplying an estimate of the total number of record you are about to have SORTed. If you also enter INCLUDE and DELETE statements, SRTUTL will calculate this number. Otherwise, SORT will attempt to determine file sizes from disk or diskpack headers. On the other hand, if the input is form tape and the SORTing media is not tape, you must enter the RECORDS statement.

The syntax of the RECORDS statement is shown in Figure 19-11 below.

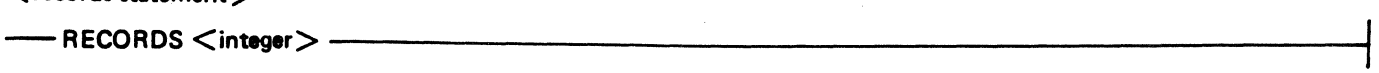


Figure 19-11. Records Statement

Execution Instructions

You can execute SRTUTL using a card deck as shown in Figure 19-12 below. Alternatively, you can execute SRTUTL without cards by using an INSERT statement as shown in Figure 19-13 below.

You may initiate SRTUTL with either the EXECUTE or the COMPILE command. These two commands are depicted in Figure 19-12 and 2113, respectively. The advantage of using the COMPILE command is that additional features are provided. These include:

1. In ODT displays, system logs, and WFL job summaries, the program name is represented as "user-id/SRTUTL." The user-id can be any 1 to 6-character identifier, such as the name of the file being SORTed, or the name of a program in a logical job stream. Thus, individual copies of SRTUTL in a multiprogramming mix can be identified easily.

2. If SRTUTL detects errors in the input parameter statements, you are notified, and the execution of subsequent programs can be affected.

   a. "** SYNTAX ERR" appears on the EOJ line in the ODT display, and on the EOT line in a WFL job summary.

   b. If the MCP's AFTR option is set, and if SRTUTL detects an input parameter error, programs linked to this execution by the MCP command AFTER are removed from the schedule. For example:

      ?COMPILE PA110 SRTUTL: FILE CARD PA110C
      ?EX PA120 AFTER PA110/SRTUTL

   If the system option AFTR is set, PA120 is executed only if SRTUTL finds no errors in the parameters and goes to a normal end-of-job.

   c. A WFL job may test for successful processing of the input parameters using the COMPILEDOK syntax:

      COMPILE PROG1 WITH SRTUTL [P1];
      IF P1 IS COMPILEDOK THEN...

If you initiate SRTUTL by EXECUTE (or, in WFL, RUN), errors in the input parameters are noted only in the output listing, and SRTUTL goes to a normal end-of-job. You are not notified of the errors, and subsequent programs will assume that the SORT has occurred.

```
                  { EXECUTE          }
                  { EX               }
?                 {                  }   SRTUTL
                  { COMPILE user-id [WITH] }
                  { CMP              }
?DATA CARD
    program statements
?END
```

Figure 19-12. SRTUTL Execution Card Deck

```
                  { EXECUTE          }
                  { EX               }
?                 {                  }   SRTUTL IN 0 4 UA "abcd"
                  { COMPILE user-id [WITH] }
                  { CMP              }
```

Figure 19-13. SRTUTL Execution Without Cards

SRTUTL, by default, expects a card file as input. This default may be overridden by adding the INSERT clause to the EXECUTE command.

SRTUTL allocates dynamic memory to hold buffers, and includes and deletes code. Normally, the space is adequate; however, if the files to and other output from the SORT have very large blocks, you may need to enter a ?MEM =nn statement.

Cardless Executions

In the INSERT clause, a, b, c, and d represent alphabetic character positions. You must fill each position with a letter or a blank, and you must enclose string in quotation marks. You must always leave position c blank.

Each position within the INSERT clause represents an input file or file format. The character in the position indicates the device type on which the file is located, or it indicates the format of the file as shown in Table 19-1 below.

Table 19-1. Valid INSERT Values for Cardless Execution

Position	Represents	Permissible	Default
a	Primary Symbolic File	C, D, P, T, s	C
b	Secondary Symbolic File	C, D, P, T, s	D
c	Not Used	s	s
d	Primary File is Editor Format	E, s	s

NOTE
The character "s" represents a space.

Table 19-2. Meanings of INSERT Values for Cardless Execution

Position	Character	Meaning
a, b	C	Card Reader
	D	Disk (any 100-byte)
	P	Disk diskpack (180-byte)
	T	Magnetic Tape
	s	device unspecified
d	E	EDITOR-format primary symbolic file
	s	Primary symbolic is non-EDITOR file

NOTE
The character "s" represents a space.

You can indicate the input files by specifying the appropriate combination of permissible values from the second table in positions a, b, and d of the INSERT clause. As soon as you have specified the media and file format, you can assign file names on dollar records or in FILE equate clauses if the actual names of the files are other than the default names assumed by SRTUTL.

Table 19-3 summarizes the input and output files and formats that are acceptable when you use SRTUTL (an X indicates an acceptable file), SRTUTL's internal name for each file (for use in FILE equating), and the values that must be supplied in the INSERT clause when executing without cards.

Table 19-3. SRTUTL Files and Format

Input/ Output File	File name	80x1 card file	80x disk file	EDIT file on disk	80x disk pack File	Disk-pack edit file	80x tape file	132x1 print file
Primary symbolic	CARD or EDITFL	X	X	X	X	X	X	
Insert value		(a=C)	(a=D)	(a=D, d=E)	(a=P)	(a=P, d=E)	(a=T)	
Secondary symbolic	SOURCE	X	X		X	X		
Insert Value		(b=C)	(b=D)		(b=P)		(b=T)	
Output Listing	PRINT						X	

NOTE

Source input files may be any blocking factor from 1 through 9. The name of the primary file is CARD when it is a non- EDITOR-format file, and EDITFL when it is an EDITOR file.

DISK is the default medium for the primary symbolic file when you specify EDITOR format.

The values you specify in an INSERT clause take precedence over information in dollar records. For instance, a dollar record may supplement the information in an INSERT statement by providing the name of the secondary input file.

Below are two examples of execute statements for cardless executions:

EXECUTE SRTUTL IN IN 0 1 UA "D" FILE CARD = PROG1S

Input specifications for SRTUTL are in a disk file named PROG1S. There is no other input file.

EXECUTE SRTUTL IN 0 4 UA "DP E" FILE EDITFL = PATCjc

There are two input files containing SRTUTL input specifications. The primary symbolic file is an EDITOR file on disk, named PATCjc. It contains records to be added to other records or to replace other records in the secondary file. The secondary symbolic file is on disk pack; its name and the name of its diskpack are specified on a dollar record (not shown).

Label Equation

It is necessary to use label equation cards to uniquely identify source program files in a multi-programming environment. The internal file-names and external file identifiers for SRTUTL files are shown in Table 19-4 below.

Table 19-4. SRTUTL File Names

Internal File-name	External File-ID	Function
CARD	CARD	Non-EDITOR primary symbolic file
EDITFL	EDITFL	EDITOR Primary symbolic file
SOURCE	SOURCE	Secondary symbolic file
PRINT	PRINT	Printed output listing.
PDATAB	PmxTpB	SRTUTL work file, where mx indicates mix- number and p is the processor-number

When you want to change the external file-identifier on a primary source language file, the label equation statement you would use is:

?FILE CARD = <file-id>

where <file-id> is a unique file-identifier of six characters or less.

You can also direct files to specific hardware devices through the use of the MCP FILE statement. For example, to direct the SRTUTL work file to a specific diskpack, you can use the following label equation:

?FILE PDATAB = pack-id/PDATAB DPK

where diskpack-id is the name of the diskpack you want. Note that by default the SRTUTL work file resides on the same hardware device as the output SORTed file.

You can specify punched cards, disk, diskpack, or magnetic tape as a source language input medium for a single input file or for a master file. If you use two input files, SRTUTL merges them according to sequence number.

Records of input files are 80 bytes in length. You may use any blocking factor between 1 and 9 inclusive for input from disk, diskpack, or magnetic tape.

Dollar Options

Certain options are available to you during operation, and you can activate or deactivate them by what are referred to as "dollar" options. These options have a dollar sign character ($) in column 1, and may be interspersed at any point in or immediately before the primary or secondary input file. One or more dollar options may be specified following the $.

If you do not have control records, the default entry is LIST, and a single card deck is expect as input.

The dollar options SET and RESET let you turn on or turn off options you specify on a given line, without affecting any other options you do not specify. When SET or RESET is not the first $ option listed on the $ card, the options listed are turned on and all other are turned off.

The dollar options available to you are listed immediately below, along with a brief description.

CHECK

Sequence errors are flagged with a warning message.

CLEAR

All options that do not have associated values are reset, except MERGE.

CORRECTOK

When you set CORRECTOK in a dollar record, SRTUTL will attempt to correct certain syntax errors in the SORT parameters. When SRTUTL detects an error, which it can attempt to correct, the error is flagged with a message in the following form:

CORRECTED: <text of error message>

The word CORRECTED is printed when a correctable error is detected, regardless of whether the correction has actually been made. However, the correction is not actually made unless you have specified $CORRECTOK.

DELETE

This option may appear only in the primary source file, and is ignored unless you set $MERGE. Images from the secondary source file are discarded (including other dollar options) until you reset this option. Items discarded are not listed in the execution listing unless the LISTDELETED option is set.

DOUBLE

Double-spaced listing of source images processed is produced, along with error messages.

ERRORLIMIT = nnn

Execution is terminated when the number of errors you specified in nnn is reached. The nnn must be a 3-digit integer. It has a default value of 100.

LIST or LIST$

Single-spaced listing of source images processed is produced. This option is assumed unless you reset.

LISTCCI

Lists are produced of all dollar options in the output listing. Compiler Control Images (CCI) is another term for dollar options.

LISTDELETED

Causes deleted or voided images to be listed in the output.

LISTOMITTED

Causes omitted images to be listed in the output.

LISTP

Lists card images only (not source file) when you reset LIST.

MERGE ["file-identifier"] [device]

Allows compilation of a program using two symbolic input files. The inputs are merged on the basis of sequence numbers, one input being the primary input file that contains this option. The file-identifier is either "file id" or "multi-file-id/file-id", where file-id and multi file-id must be alphanumeric literals, each not more than six characters long. File-id specifies the file-name of the secondary input file. SOURCE is the assumed file-name if none is specified; however, this file-id may be entirely overridden by label equation. Device is the hardware on which the second input file resides. Valid devices are DISK, TAPE, and diskpack with DISK as the default.

Once you set this option, you cannot reset it and it remains in effect for the entire program.

OMIT

This option may appear in both primary and secondary source input files. Images form any source are ignored (but passed to new source) until you reset this option. Items omitted are not included in the output listing unless you set the LISTOMITTED option. Any dollar options encountered while OMIT is set are processed normally.

PAGESIZE = nn

Specifies that the maximum number of lines for each page of the output listing is to be equal to nn. The nn must be a 2-digit integer. The default value is 56.

SEQ nnnnnn +nnnnnn

Source images are resequenced starting with the sequence number nnnnnn and incrementing by +nnnnnn. The default start (or base) is 10, and the increment defaults to +10.

SEQCHECK

Same as the CHECK option above.

SEQUENCE

Same as the SEQ option above.

SUMMARY

Causes a listing of the execution summary when you reset the LIST option. This summary is included in the LIST option.

VOID nnnnnn

Images from secondary source file are discarded until a sequence number larger than nnnnnn is encountered in the secondary source file.

SECTION 20
AUTOMATIC SYSTEM RECOVERY FACILITY (SYSUP)

The SYSUP facility provides a means to automatically recreate the operating environment after a haltload or system failure. The facility is specifically advantageous for systems that fail in an unattended, on line environment. In the event the system fails, the SYSUP facility can provide information regarding the type of haltload, and the task number of the job being executed at the time the system failed. It can also execute a program written by you that goes about such tasks as executing programs, readying printers, and aspects of recreating the operating environment.

The MCP can automatically initiate the SYSUP facility after any type of haltload. If a version of SYSUP written by you is available, the MCP executes it. Otherwise, it executes a version bound to the MCP.

The MCP passes two digits of information to the SYSUP program when it is initiated. The first digit describes the type of haltload that took place, the second digit contains the processor number.

The data passed to SYSUP by the MCP can be used by a SYSUP program written at your installation. These version tend to be more sophisticated than the version bound to the MCP, since they are written to suit the needs of the installation. The bound version is fairly simple. It zips a command to the MCP and goes to EOJ (end of job). On the other hand, other versions of SYSUP do such things as ready diskpacks, set up printers, dump MCP memory, and bring up time-sharing and handlers.

A haltload takes place when the system operator inputs the maintenance processor command LOAD MCP. An automatic haltload takes place after the system fails if the USE AUHL option is set. The SYSUP facility is executed automatically if the USE SYSUP option is set. For more information regarding the USE SYSUP and USE AUHL options, see the *V Series MCP/VS System Software Operations Guide, Volume 1: Installation*

When executed, the bound version of SYSUP sends a command to the MCP to fire up DMPANL to analyze the MCP memory dump file.

```
PM 1
```

The SYSUP facility stores two digits of information in memory at base-relative addresses 32-33. The digits indicate the type of halt load, and the processor number.

The first digit indicates the type of haltload. It can have the following values and meanings:

Digit	Type of haltload
0	manual haltload
1	ODT haltload
4	coldstart haltload
9	automatic haltload

The second digit indicates the processor number.

SYSUP Programming Considerations

A full dump is usually required in order to learn why the system failed. Therefore, it may be a good idea to have the version of SYSUP you write execute DMPANL by zipping a PM1 request. Then wait for

DMPANL to finish what it is doing before you fire up any jobs or handlers. You might want to print out the MCP table (TBL) dump for immediate use or copy the dump file ($p0001) to tape or diskpack for later analysis.

Example SYSUP Programs

Two example SYSUP programs follow. The first is the bound SYSUP program; the second is an example of a user SYSUP program designed to restart an unattended time-sharing system.

```
BEGIN
  CONTROL STACK:= 0& FLAT PROGRAM
  INTEGER WHOCALLS (1) = 32,
          PROCNUM  (2) = 33;
  ALPHA DUMPER (16) := "SPO PM1"
  IF WHOCALLS = 9 THEN
  BEGIN
    ZIP DUMPER;
  END;
END;
```

Example 2:

```
SYSUP:
BEGIN
  CONTROL STACK := 0;& FLAT PROGRAM;

  INTEGER WHOCALLS (1) = 32,
        PROCNUM (1) = 33,
        PROGRAM#RUNNING (2);& TO SEE IF JOB IS RUNNING;

  ALPHA DUMPER (8) := SPO PM1;

  & FIRST READY ALL OF THE DISKPACKS & IN CASE H/L SAVED THEM;
  ZIP SPO RY 8/0.;
  ZIP SPO RY 8/1.;
  ZIP SPO RY 8/2.;
  ZIP SPO RY 8/3.;

  & NOW SET UP THE PRINTERS CORRECTLY;
  ZIP SPO RY7/0 DOC96.; & WHITE PAPER;
  ZIP SPO RY 17/0 SHORT.; & SHORT BLUE PAPER;
  ZIP SPO RY 15/0 LINED.: & GREEN STOCK FORMS;
  ZIP SPO RY 19/0 SPEC.;& DOCUMENTATION PRINTER;

  & SAVE THE SPECIAL FORMS PRINTERS;
  ZIP SPO SV 17/0.;
  ZIP SPO SV 15/0.;
  ZIP SPO SV 19/0.;

  & IF AUTOMATIC HALTLOAD DO A DUMP FIRST;

IF WHOCALLS = 9 THEN& AUTOMATIC H/L?
BEGIN
  ZIP DUMPER;
  & WAIT FOR DMPANL TO FINISH;
  DO BEGIN
    DOZE 10;
    PROGRAM#RUNNING := TASKID DMPANL;
  END UNTIL PROGRAM#RUNNING = 0;
END;& END OF SYSTEM DUMP CODE;

  & BRING UP THE SYSTEM MONITOR (IF NOT ALREADY UP)

  PROGRAM#RUNNING := TASKID SYSMON;
  IF PROGRAM#RUNNING EQL 0 THEN
    ZIP EX SYSMON PR 6 MEM 13 LOCK.;

  & AND NOW TO BRING UP TIME-SHARING;

  ZIP EX CTLMCS PR 7 LOCK.;
END;
```

SECTION 21
UNILINE DLP UTILITY (UNLODV)

Uniline DLPs connect the processor to different kinds of data terminal equipment (DTE) and data communications equipment, including OCSs, remote job entry terminals and teletypewriters. Other devices include various kinds of RS232 equipment and modems.

UNLODV matches the data communications parameters used by the Uniline DLP to those used by the device that is attached to that Uniline DLP. It sets these data communications parameters in the form of firmware, and can either save the firmware in a file, or load it to the DLP. If saved in a file, the firmware can be loaded to the Uniline DLP at a later point in time with the MCP command LH, or the LOADFW utility or UNLODV.

The data communication parameters must be matched in order to establish the first level interface, the electrical connection used to exchange and check signals between the Uniline DLP and the device. If these parameters are not matched up, the chances are high that the Uniline DLP and the device will not communicate.

There are two different files that store the basic firmware that UNLODV reads in and modifies. The files are named USP2BH and UST2BH. Which file you use depends on what classification of equipment is attached to the Uniline DLP. These basic firmware files are distributed on the system software release tape and on the UNILINE support files tape, which is included in the UNILINE test and field package.

UNLODV customizes the firmware in memory in accordance to the commands you input. Then, depending on what you tell it to do, UNLODV either loads the firmware into the Uniline DLP, or it stores the customized firmware in a file named by you or both. This file can be loaded into the Uniline DLP at a later time.

Which of the two basic firmware files you use depends on what classification of equipment is attached to the Uniline DLP:

Firmware File USP2BH

This file contains SPO look-a-like firmware. Use this file if the device connected to the Uniline supports a secondary ODT connected via an RS232 asynchronous direct connection. Some of the equipment this supports includes an ET, TD8, MT983, and a TC 4000.

USP2BH Settings:

ASCII	SET
BAUD RATE	09600
ADAPTER-IO	OO
ASYNCHRONOUS	SET
EIA	SET
83900	SET
UNILINE	2AC
CHAR-LENGTH	7 BITS
DIRECT CONNECT	SET
TIMEOUT-EOT	RESET
RETURN CR AS:	CR
RECEIVE-DELAY	00001
TRANSMIT-DELAY	00100
SQUELCH-DELAY	00000
TIME-CUT	09999
STOPBITS	1

Firmware File UST2BH

This file contains standard terminal control look-a-like firmware used for various types of data terminal equipment (DTE) and modems. It can be used to connect the Uniline to a network of terminals using poll/select, or to communicate system-to-system either via a direct connect or a telephone system. It runs RS232, synchronous or asynchronous, or TDI, with or without modems. The connection can be via switched (dial) or leased lines.

UST2BH Settings:

ASCII	**SET**
BAUD RATE	09600
ADAPTER-IO	OO
ASYNCHRONOUS	SET
EIA	SET
83900	SET
UNILINE	2AC
CHAR-LENGTH	7 BITS
DIRECT CONNECT	SET
TIMEOUT-EOT	RESET
RETURN CR AS:	CR
RECEIVE-DELAY	00001
TRANSMIT-DELAY	00002
SQUELCH-DELAY	00016
TIME-CUT	05000
STOPBITS	1

You can display the setting of the data communications parameters in a file. First you must execute UNLODV and load the file in memory. Then input the DISPLAYPARA command to display the settings on the screen.

MCP Configuration Records

Two different types of MCP configuration records are used to declare Uniline DLPs. A DLP record must be used to declare what channel the Uniline DLP is on. Then a UNIT record must be used to declare the type of device that is connected to the Uniline DLP. The following system configuration records declare Uniline DLPs and devices to the MCP:

- DLP record
- UNIT RJE record

- UNIT TC5 record
- UNIT VDD record
- UNIT TWX record

For the syntax of these records, see volume one of this manual.

UNLODV Program Flow

You can execute UNLODV from an ODT or from a card reader. If it is executed from an ODT, the commands and data communications parameters are input using the MCP command AX. Only one command can be input per AX command.

If it is executed from a card reader, the commands and the data communications parameters must be input at the same time it is executed.

Once executed, UNLODV can customize firmware for one Uniline DLP while it loads firmware to another Uniline DLP.

Executing UNLODV

UNLODV can be executed and controlled from an ODT or through a card data reader.

Use the following syntax to execute UNLODV from an ODT:

```
EX UNLODV/<name of the file with the basic firmware>/S
```

If you execute UNLODV from a card reader, only one command or parameter can be input per record. The program expects a data card with the name COMFIL.

For example, you may use the following syntax to execute UNLODV with commands coming from a card reader:

```
?EX UNLODV/<name of file with the basic firmware>
?DATA COMFIL
TD8
MAKE FILEAA
?END
```

Many of the data communications parameters have value ranges. You must enter a value for each parameter that has a value range.

After you execute UNLODV, get its mix number. You need the mix number in order to input data communications parameters and commands to the program. The following is an example of the syntax used to AX commands and parameters to the program:

```
<mix number> AX "TD8"
<mix number> AX "MAKE FILEAA"
```

Data Communications Parameters

This table lists the data communications parameters that can be input, and any values. A value must be entered where a value appears. The default setting for each file is documented under the heading TD8.

Data Communications Parameters	Value Range
ADAPTERID	<hex number 0-1F>
ASCII	
ASYNCHRONOUS	
BAUD	<110-19200>
BDI	
CHARACTERLENGTH	<5-8>
CONVERTCR	
DATASET	
DIRECTCONNECT	
EBCDIC	
EIA	
LEASEDLINE	
PARITY	<ODD, EVEN or DISABLED>
POLLTIMEOUTEOT	<SET or RESET>
PUSH	<0-4>
RECEIVEDELAY	<0-255>
SQUELCHDELAY	<0-255>
STOPBITS	<1, 1.5, or 2>
SWITCHEDLINE	
SYNCHRONOUS	<SINGLE or DOUBLE or none>
TD8	
TDI	
TIMEOUT	<number of milliseconds 1-9999>
TRANSMITDELAY	<number of milliseconds 0-255>
U2AC	
U2B	

Note that RECEIVEDELAY and SQUELCHDELAY are additive on any I/O (including poll and transparent options) that contains a WRITE-FLIP READ.

You can shorten any parameter so long as the abbreviation is unique from other parameters. For example, you may abbreviate ASYNCHRONOUS to ASY but not to AS, because AS could also denote ASCII.

ADAPTERID <hex number 0-1F>

This sets the adapter-id in a standard terminal control. The adapter-id is used only in the result descriptor of a TEST I/O operation as is generally used by remote job applications (RJE).

ASCII

This makes the Uniline sensitive to ASCII control codes. Note that this option must match the ASCII/EBCDIC hardware strap.

ASYNCHRONOUS

This makes the transmission/reception of data asynchronous.

BAUD <110-19200>

This sets the BAUD rate for the transmission and reception of data. It is used only when ASYNCHRONOUS is set.

The BAUD rates for the Uniline type 2B range from 110 to 19200.

The BAUD rates for the Uniline types 2, 2A, and 2C range from 110 to 9600.

BDI

This enables BDI interface.

CHARACTERLENGTH <5-8>

This sets the number of data bits per transmitted and received character.

CONVERTCR

This causes a received CR code (0D hex) to be converted to an ETX code (03 hex) during a READ CONTROL operation, during the read portion of a WRITE FLIP TO READ CONTROL operation, or during the positive response to a POLL operation.

DATASET

This describes the remote device as being connected to the Uniline through a data set to a switched (dialed) line.

DIRECTCONNECT

This describes the remote device as being connected directly to the Uniline DLP: for example, without an intervening dataset. TDI or BDI also may be specified separately.

EBCDIC

This makes the Uniline sensitive to EBCDIC control codes. This option must match the ASCII/EBCDIC hardware strap.

EIA

This is synonymous with RS232.

LEASEDLINE

This indicates that the Uniline is attached to a leased line dataset.

PARITY <ODD, EVEN or DISABLED>

This describes the parity generation and checking for each character transmitted and received on the data communications line.

POLLTIMEOUTEOT

This causes a timeout on a RECIRCULATING POLL I/O OPERATION to be treated as a negative response instead of returning a timeout result descriptor.

PUSH <0-4>

The number of push characters (FF hex) appended to each synchronous transmission. Used to "push" the last data characters through the sync datasets.

RECEIVEDELAY <0-255>

This sets the delay in milliseconds between the detection of the carrier detect signal and the beginning of the firmware search for valid data.

SQUELCHDELAY <0-255>

This sets the delay in milliseconds between the last bit being transmitted and anything else happening. This value applies only to a DATASET device. Set it to zero for a directconnect line.

STOPBITS <1, 1.5, or 2>

Set the number of stopbits transmitted or expected for each character transmitted or received. Use only for asynchronous transmissions.

SWITCHEDLINE

This has the same meaning as the DATASET command and indicates that the UNILINE is attached to a switched (dial) line dataset.

SYNCHRONOUS <SINGLE, DOUBLE or none>

This indicates the transmission or reception of data as being synchronous, and the number of sync characters to expect before synchronization is achieved. Since sync characters are always expected on a synchronous line, use "SYNCHRONOUS DOUBLE" is recommended.

TD8

This sets the parameters of the firmware as follows. For Uniline 2, use command TD8 and Baud to set Baud rate.

	ASCII	
	ASYNCHRONOUS	
	BAUD	9600
	CHARACTERLENGTH	7
	DIRECTCONNECT	
	PARITY	EVEN
	RECEIVEDELAY	2
	TRANSMITDELAY	10
	STOPBITS	1
	TIMEOUT	5000

USP2BH Settings with TD8:

	ASCII	SET
	BAUD RATE	05907
	ADAPTER-IO	OO
	ASYNCHRONOUS	SET
	EIA	SET
	83900	SET
	UNILINE	2AC
	CHAR-LENGTH	7 BITS
	DIRECT CONNECT	SET
	TIMEOUT-EOT	RESET
	RETURN CR AS:	CR
	RECEIVE-DELAY	00002
	TRANSMIT-DELAY	00010
	SQUELCH-DELAY	00000
	TIME-CUT	05000
	STOPBITS	1

TDI

This enable two-wire direct interface hardware.

TIMEOUT <1 - 9999>

Set the number of milliseconds to wait for reception of a character before timing out.

TRANSMITDELAY <number of milliseconds 0-255>

This sets the number of milliseconds to delay between the reception of the clear to send signal and the transmission of data.

U2AC

Sets the firmware to that used by the Uniline types 2A or 2C.

USP2BH Settings with U2AC:

ASCII	SET
BAUD RATE	09600
ADAPTER-IO	OO
ASYNCHRONOUS	SET
EIA	SET
83900	SET
UNILINE	2AC
CHAR-LENGTH	7 BITS
DIRECT CONNECT	SET
TIMEOUT-EOT	RESET
RETURN CR AS:	CR
RECEIVE-DELAY	00001
TRANSMIT-DELAY	00100
SQUELCH-DELAY	00000
TIME-CUT	09999
STOPBITS	1

U2B

Sets the firmware to that used by the Uniline type 2B.

USP2BH Settings with U2B:

ASCII	SET
BAUD RATE	09600
ADAPTER-IO	OO
ASYNCHRONOUS	SET
EIA	SET
83900	SET
UNILINE	2B
CHAR-LENGTH	7 BITS
DIRECT CONNECT	SET
TIMEOUT-EOT	RESET
RETURN CR AS:	CR
RECEIVE-DELAY	00001
TRANSMIT-DELAY	00100
SQUELCH-DELAY	00000
TIME-CUT	09999
STOPBITS	1

UNLODV Commands

After you execute the program and input the data communications parameters to modify the firmware, you can use the following commands to save the customized firmware in a file, to load the firmware to a Uniline DLP, and to patch the firmware's object code.

END

This ends the UNLODV session.

DISPLAYPARA

This displays the data communications settings of the firmware file.

LOAD <Uniline DLP channel number>

This loads the modified firmware from memory to the Uniline DLP that is connected to the specified channel.

MAKE <file name>

This tells UNLODV to store the firmware which has been customized in memory in a file with the given name. This file can later be loaded to the Uniline DLP with the MCP command LH, UNLODV, or LOADFW.

PATCH

This allows modifications to the actual object code in the firmware. The syntax for PATCH is:

PATCH V<version number> <hex address> <hex data>

The data can be up to 60 characters. For example, if the firmware has the hex data "0123456789" patched into it at hex address 0FA8 the input command is as follows:

PATCH V601 0FA8 0123456789

Firmware File Format

UNLODV stores firmware in a single area firmware file blocked 100 bytes by 1. The first record contains information on the size of each load phase. There can be up to three load phases, although only the first phase is applicable to the Uniline.

The following table represents the first record in the file. It specifies the size of each phase. Zero indicates that the phase is skipped.

Offset	Size/Type	Description
0	6 UN	Phase One Load Size
6	6 UN	Phase Two Load Size (Zero for this program)
12	6 UN	Phase Three Load Size (Zero for this program)
18	182 UN	Zero Fill

All other records in the file contain 100 bytes of load data each. The file resides on disk.

SECTION 22
MCP MEMORY DUMP ANALYSIS PROGRAM (DMPANL)

The DMPANL program reads an MCP memory dump from disk, diskpack, or tape, and produces a printer backup file containing a formatted analysis of the dump. The internal name of the printer backup file is MEMORY/DUMP.

REQUIREMENTS FOR DUMP ANALYSIS

The following requirements must be met before you can use DMPANL:

The coldstart deck (or the system configuration file) must include a USE record specifying the dump file location. For example:

```
USE DUMP DISK
```

If there is no USE DUMP record, the only way to obtain a memory dump is with the DMPMEM utility. Refer to the *V Series System Software Operations Guide Volume 1: Installation* for information about the USE DUMP record.

The MCP must be linked using the standard MCP LINKER template. DMPANL requires that all link decks contain the same number of modules in the same relative order.

The version of the MCP MID that is used to link an MCP should be compatible with the versions of DMPANL and DMPOUT.

The XM firmware should not be patched to halt on hardware calls. Such patches prevent entry to the fault handler, which produces the dump.

STARTING DMPANL UNDER MCP/VS 2.N

To analyze an MCP/VS 2.n dump on a machine that is running MCP/VS 2.n, use the system command PM 1 to initiate DMPANL.

You can enter the DMPANL command syntax when you enter the command PM 1, by stringing the commands after the PM 1. For example:

PM 1 ALL

Or, you can enter the command followed by an asterisk. The asterisk makes DMPANL wait for commands in the form of <mix number> AX (accept) messages. If you use AX messages, you must terminate the string of commands with the word END. For example:

You enter: PM * 1

The system responds: ** $00001/DMPANL=05 ACCEPT

You enter: 5 AX ALL END

Commands to DMPANL are additive; that is, if you request a given table anywhere in the DMPANL commands it will be printed.

Text Deleted by PCN-001.

DMPANL COMMAND SYNTAX

Because of the ever-increasing size of the MCP tables, it will become more and more important for you to limit the information that you want to print. The command syntax enables you to specify what is printed, down to the level of which specific tables you want. The syntax appears later in this section.

If you enter the command PM 1 with no parameters, the results depend on how the USE DUMP record is set in the system configuration file. If the USE DUMP record specifies DISK or PACK, DMPANL produces a brief summary listing and copies the entire memory dump to a tape. If the USE DUMP record specifies TAPE, DMPANL produces an error message because no memory dump file is available.

If you enter an asterisk, DMPANL displays the following message:

ENTER <DUMP SPECS> OR HELP OR END

DMPANL then issues an ACCEPT and awaits input. You respond as follows:

<mix no> AX <dmpanl parameters>

If you use the asterisk (*) and enter AX commands, then:

You can enter multi-line input.

You must enter END to finish the DMPANL command syntax.

You can enter <mix no> AX HELP to get a list of DMPANL parameters. DMPANL then awaits input.

If the system detects a syntax error in the parameters, the system displays an error message and ignores all parameters following the one in error. DMPANL then continues to await input.

Figures 22-1 through 22-8 show the syntax diagrams for DMPANL. Explanations of the key words follow each diagram.

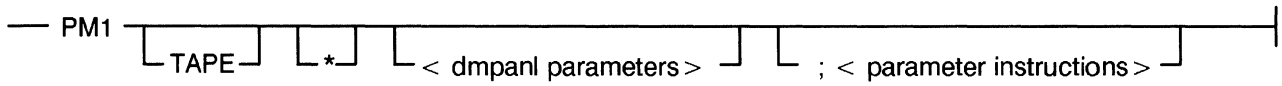


Figure 22-1. PM Command Syntax for MCP Dumps

TAPE

The source of the memory dump is tape. The tape could have been created with the DMPMEM utility, or by DMPANL as described in this section. If no other parameters are given, DMPANL produces a brief summary listing.

*

If you enter an asterisk, DMPANL will issue an ACCEPT and await input. You can enter multi-line input this way.

<parameter instructions>

These are system commands such as CHARGE, AFTER, MEMORY, and LOCK.

<dmpanl parameters>

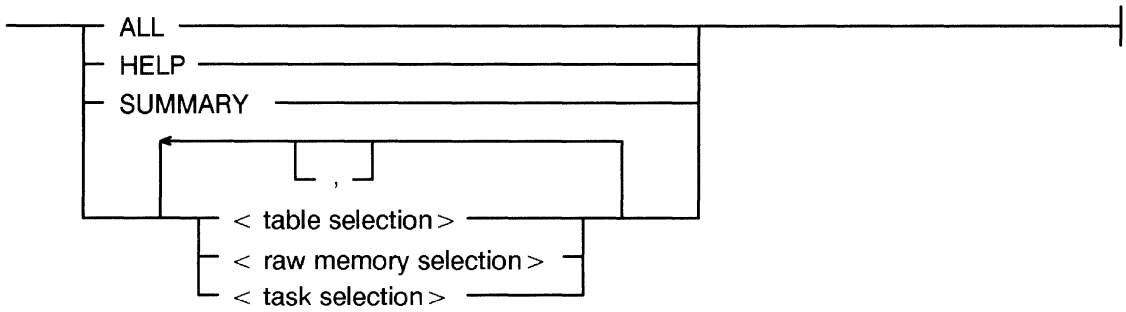


Figure 22-2. DMPANL Command Parameters

ALL

Prints all of the loaded MCP, the system tables, and all other allocated memory.

In most cases, the listing produced by the ALL option will exceed the maximum disk or diskpack printer backup file size. File-equate the PRINT file to tape.

HELP

This produces a list of the DMPANL parameters. It can be entered only after an asterisk (*). After producing the list, DMPANL continues to await input.

SUMMARY

Prints a brief summary of the dump, including the current task information, the function path, and the current task MCP stack frame.

<table selection>

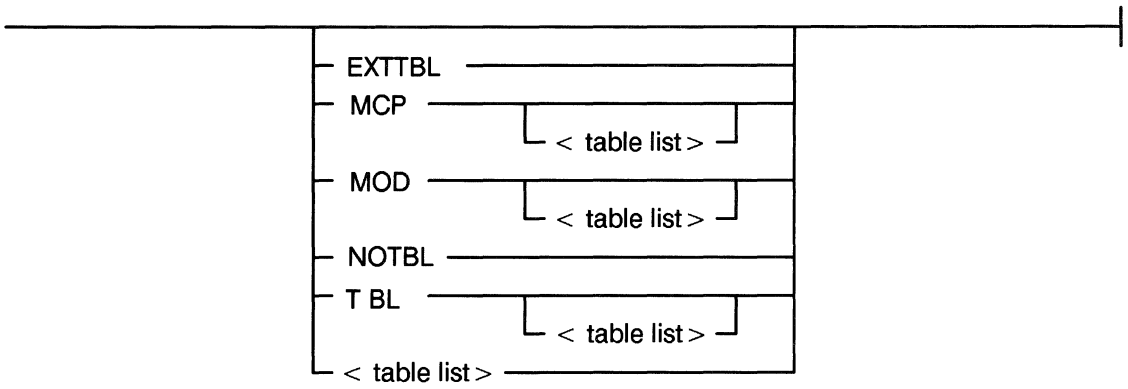


Figure 22-3. DMPANL Table Selection Options

EXTTBL

Prints the system tables and all of the task MCP tables such as the soft IOATs, the incore DFHDRs, and the address blocks. No raw memory dump is produced.

MCP

Produces a system table breakout and a raw dump of the MCP and overlay area.

In most cases, the listing produced by the MCP option will exceed the maximum disk or diskpack printer backup file size. File-equate the PRINT file to tape.

MOD

Prints the system tables, the memory from 0 KD to the top of the overlay area, any MCP modules that were loaded, and any independent runners that were executing.

In most cases, the listing produced by the MOD option will exceed the maximum disk or diskpack printer backup file size. File-equate the PRINT file to tape.

NOTBL

Overrides the default of printing the system tables when specific MCP tables are requested. TBL

Prints the system tables, but no raw memory dump.

<table list>

The internal operating system data structures (tables) that can be analyzed by DMPANL are listed on the following pages.

Figure 22-4 Deleted by PCN-001.

CMPLX

Prints information about tasks waiting in complex wait.

DCP

Prints all of the information relating to the data communication processors. This includes the DCP Station Table, and the DCP and MCS tables, including their entries and buffers.

DMS

Prints information from two internal DMSII data structures; the Database Program (DBP) Table and the Database User Program Control Table.

EU

Prints the information in the EU table and in the disk subsystem table.

FILE

Prints the following information for every open file:

- the FIB
- the IOAT
- the external FIB
- the incore DFHDR
- the address blocks

The FILE option also prints the following for every open PORT file:

- The Port Attribute Structure
- The Subport Directory
- The Subport Attribute Structure

GLOBAL

Prints the global code and overlay segment.

HLPARMS

Prints the haltload parameters.

IOAT

Prints the information in the IOAT table.

IOQUE

Prints the information about each channel and about the queue elements that are in process and waiting to be fired.

JMR

Prints internal data structures used by the Job Manager.

MAST

Prints the Memory Area Status Table and prints where memory is available on the system.

MAT

Prints the Memory Area Tables for every task and MCP function.

MIX

Prints the MIX table.

MSG

Prints the messages that are currently waiting in the Message Module's message pool.

PORTS

Prints the information in the Port Global Data Memory Area and the Candidate List memory area.

The Port Global Data Memory Area contains entries for active ports on the system. Each entry contains attributes shared by all subports connected to the port, including an event directory.

The Candidate List memory area contains entries for subports the system is trying to open. An entry contains attributes needed to match subports, control state and status information.

SEC

Prints the security user table.

SEG

Prints the segment dictionary table.

SLOG

Prints the last hundred records of the ODT log.

SNAP

Prints a formatted listing of the SNAP picture contained in the memory dump file. SNAP pictures are inserted into memory under certain program or system failure conditions. The contents of a SNAP picture differs depending on the type of V series system (V 300, V 500 and so forth).

SUBPORTIO

Prints the information in the Shared Subport Memory Area (SSMA). There is one SSMA for each pair of open Subports on the system. An area contains the input and output data queues and attributes shared by both connected Subports.

TRAK

Prints the TRAK buffer.

TS

Prints the Time Sharing Processor Information Block and the Shared Area Table.

VCS

Prints the internal data structures used by the V Series Communications System (VCS).

WAIT

Prints the information about every task waiting on the system, including the locks it owns and the locks it is attempting to get.

<raw memory selection>

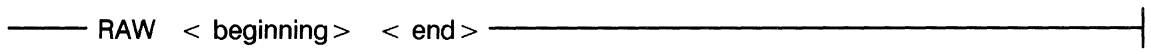


Figure 22-5. DMPANL Raw Memory Selection Syntax

RAW

Prints memory from the specified beginning address to the specified end address. No system tables are printed.

<task selection>

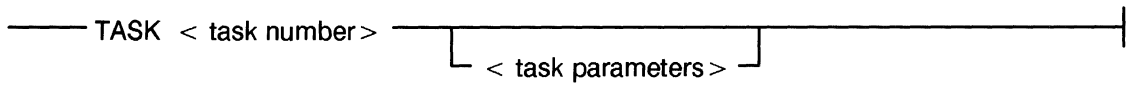


Figure 22-6. DMPANL Task Selection Syntax

TASK

Prints information about the task, which is indicated by the number that follows the command. The default is all of the task's memory areas and MCP data relating to the task. If you select TASK, you can specify the <task parameters> that appear in Figure 22-7.

<task parameters>

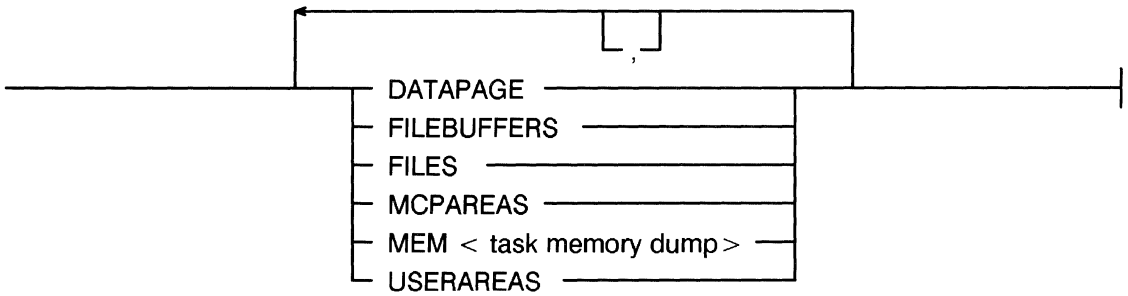


Figure 22-7. DMPANL Task Selection Parameters

DATAPAGE

Prints that task's MCP data page.

FILEBUFFERS

Prints the same information as FILES, and also prints the external file buffers of any files that have them.

FILES

Prints the information about all of the files that are open or that were opened and closed without release.

MCPAREAS

Prints just the MCP data related to that task.

MEM

Prints a raw dump of the specified memory area for the task. In getting a raw memory dump of a memory area, if no MAT is specified, the MAT defaults to the first MAT. Similarly, if no memory area is specified, memory area zero is used, which is the user data area. For code files generated by pre-V Series compilers, the only allowable MAT is the default. The begin and end addresses are specified in KD; and if they are not specified, the entire memory area is printed.

USERAREAS

Prints just the user's data and code.

<task memory dump>

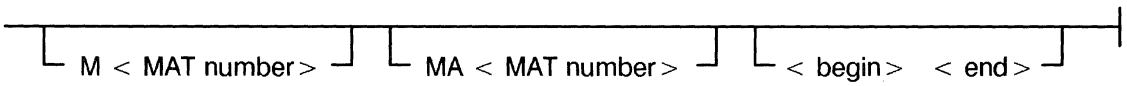


Figure 22-8. DMPANL Task Memory Selection Syntax

M <MAT number>

Indicates that the system will default to the first MAT number, if you do not specify a MAT. For code files generated before MCP Mark 1.0, the only allowable MAT is the default.

MA <memory area no>

Specifies the memory area zero, which is the user memory area, if you do not specify a memory area.

<beg>-<end>

Specifies the beginning and ending addresses in KD. If you leave this blank, the system prints the entire memory area.

See the commands DM, DP, IL, and PM in the *V Series MCP/VS System Software Operations Guide Volume 2: System Commands*.

SECTION 23
QWIK DISK

This section documents the V Series QWIK Disk, a method of extra processor memory to simulate a high performance 100-byte disk subsystem. This guide outlines the system bottlenecks that can be helped by QWIK Disk, file selection criteria, file placement suggestions, installation and coldstart procedures, as well as operational and programming considerations for QWIK Disk.

QWIK Disk is easy to use and easy to install. It can be used on any V Series processor. It is fully compatible with all releases of the V Series MCP/VS. You do not need to change any existing software to use QWIK Disk.

Because QWIK Disk simulates existing 100-byte disk subsystems, it is very easy to use. You can read and write to files on QWIK Disk in the same way you read and write files to any other 100-byte disk. To maximize your system's total performance, you must be selective when deciding what files to put on QWIK Disk. Because of its limited capacity, QWIK Disk is not recommended for use as default disk. Also, because QWIK Disk makes use of main memory, information stored in QWIK Disk can be lost due to power failures or some system faults. Critical files should be considered for QWIK Disk only if you have adequate backup procedures or if the performance benefits justify the potential recovery overhead.

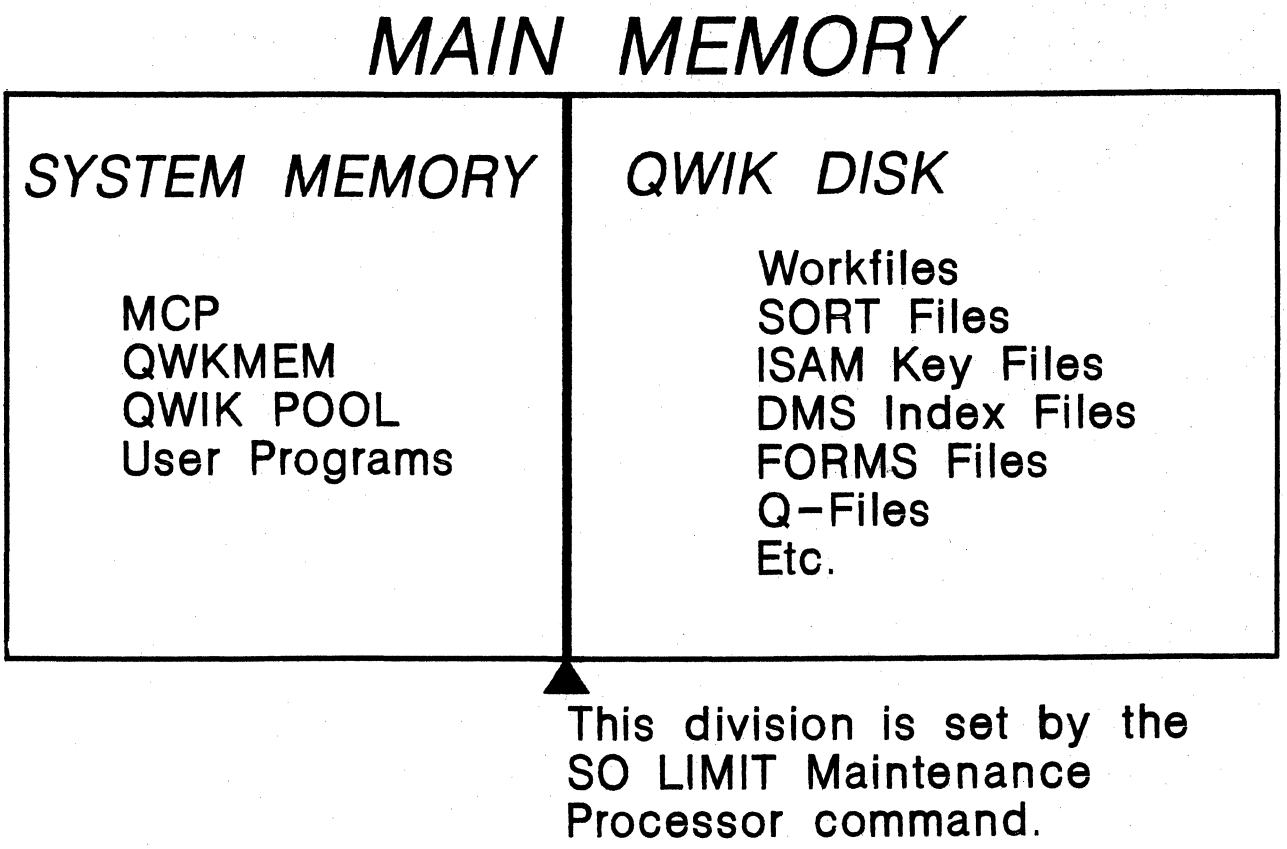
Generally speaking, QWIK Disk is an excellent storage medium for files that have a high I/O volume. It is also an excellent medium for a read-only file in a transaction processing system that requires a short response time. When choosing files for QWIK Disk, keep in mind a basic "size to I/O" ratio. Use the smallest files with the highest I/O volume.

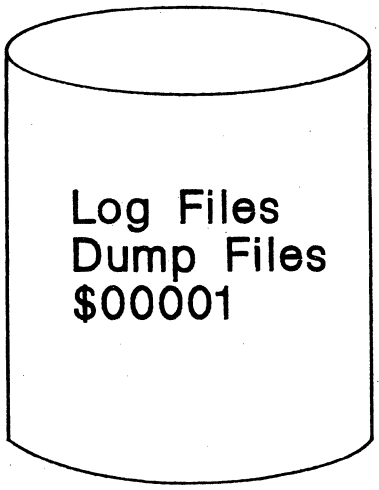
QWIK Disk significantly reduces the amount of time it takes to complete an I/O. QWIK Disk does not reduce the number of I/Os that take place. It speeds them up. It does this in two ways. First, by using RAM memory for disk storage, it eliminates "seek time", which is the time spent waiting as a disk head goes through the mechanical process of finding a sector and reading data. Second, QWIK Disk has a high data transfer rate.

To install QWIK Disk, you must:

- Load the latest IOP firmware from floppy disk.
- Determine the amount of the memory to be used by the MCP and the amount to be used by QWIK Disk.
- Use two Maintenance Processor commands to set the MCP limit and the QWIK Disk option.
- Coldstart the system with two special records in the MCP configuration file. One record tells the system how much storage QWIK Disk has. The other record reserves channel 40 for QWIK Disk.

Once installation and coldstart are complete, you can read and write files to and from QWIK Disk as you would to any other disk subsystems.





Disk

Figure 23-1. Example Memory Partition Between the MCP and QWIK Disk.

PERFORMANCE IMPROVEMENT GUIDELINES

The performance figures for QWIK Disk vary depending upon the file classes and individual files selected for QWIK Disk. Heavy peripheral activity can cause performance variance.

The following paragraphs discuss some performance guidelines based upon benchmark tests run in empty environments. For consistency, the performance improvements are expressed as "factors" computed as:

Factor = Old Time / New Time

QWIK DISK PERIPHERAL PERFORMANCE

Because QWIK Disk eliminates seek and latency time, the average QWIK Disk I/O costs 1 to 2.4 milliseconds. For QWIK Disk, I/O time is all data transfer time. QWIK Disk can transfer the largest possible block size (20,000 bytes) in 16 milliseconds.

QWIK Disk I/O speed can be at least 9 times faster than devices with movable heads, such as LAK 206 and 207.

GENERAL SYSTEM PERFORMANCE

When used in a general production or development environment, QWIK Disk performance results vary significantly depending upon which files you place in QWIK Disk. QWIK Disk improvement usually varies directly with the percentage of total I/Os shifted to QWIK Disk.

CODE FILE OVERLAYS

Most systems use the QWIKPOOL option to avoid disk overlay accesses. Doing overlays in the QWIKPOOL is advised.

SORT PERFORMANCE

Sort and SRTUTL performance can improve based upon placement of input, output, and/or work files on QWIK Disk. SORT: shows marginal improvement with work files on QWIK Disk.

SORT input and output files usually receive more I/Os than work files (which generally have large blocking). For optimum SORT performance, place at least the SORT input and work files on QWIK Disk.

DMSII AND ISAM PERFORMANCE

Due to the large number of files involved, DMSII and ISAM performance figures vary significantly.

ISAM key files and DMSII index files (sets and subsets) are good candidates for QWIK Disk because of their small size.

Functions with high I/O waiting time, such as ISAM and DMSII DELETEs and OUTPUT WRITEs (CREATE/STOREs) are particularly effective if the files involved reside on QWIK Disk.

TRANSACTION SYSTEM PERFORMANCE

Transaction-based mixes generally consist of an MCS, transaction processors, and a DMSII DBP. DMSII structures will usually receive the highest I/O traffic.

APPLYING QWIK DISK TO YOUR SYSTEM

This outlines a method of analysis that can help you isolate the types of system bottlenecks that QWIK Disk can alleviate. It also includes examples of how QWIK Disk can be used in different types of processing environments. It also includes a list of the various system reports and tools that can help you with this analysis.

Since Unisys systems are used worldwide for innumerable applications, this analysis can only be considered as an outline, a guide, something to complement your understanding of your system environment.

SYSTEM ANALYSIS - AN OVERVIEW

In order to integrate QWIK Disk into your processing environment, you need to study the following system characteristics to isolate the types of bottlenecks QWIK Disk can alleviate. These characteristics are defined below and are discussed in greater detail in the following pages.

- Processor Usage
- Memory Utilization
- Peripheral Activity

Processor Usage — Examine the processor usage to find out the MCP idle time. If the MCP idles more than 10 percent, a QWIK Disk optimization opportunity exists, and you should proceed with the rest of the analysis. If not, the system is "processor bound", and QWIK Disk can do little to improve its total performance.

Memory Utilization — After you examine the system processor usage and find out the MCP idle time, look at the system memory utilization. Do this to optimize your MCP and program overlay performance.

Find out if the MCP spends over 10 percent of its waiting time waiting for MCP overlays. If so you may be able to improve your system performance by moving the MCP code file to QWIK Disk.

Remember, QWIKPOOL is the best tool to use for overlays. If it is not already in use, it can increase your system overlay performance more than QWIK Disk.

Find out if the MCP spends over 10 percent of its waiting time waiting for program overlays. If so, you may improve system performance by moving those programs to QWIK Disk.

However, QWIK Disk can improve the performance of programs with overlays too big for the QWIKPOOL. Oversized overlays normally come from disk. But if you move the program into QWIK Disk, they will go to RAM memory instead.

Peripheral Activity — Finally, after you optimize your systems memory performance, you can begin to isolate its I/O bottlenecks.

Find out which peripheral devices have the most I/O traffic. Find out if traffic is spread out evenly among all the peripheral devices, or if it is lopsided. If so, you can improve performance by relocating files onto different peripheral devices as well as QWIK Disk.

Also look for high I/O traffic to classes of files, individual files, and individual file areas. In these cases, move the groups of files, the individual file, or the individual file areas to QWIK Disk.

When moving files to QWIK Disk keep in mind this golden rule: move the smallest files with the highest I/O volume.

SYSTEM ANALYSIS

The following pages elaborate on the system analysis described in the preceding paragraphs.

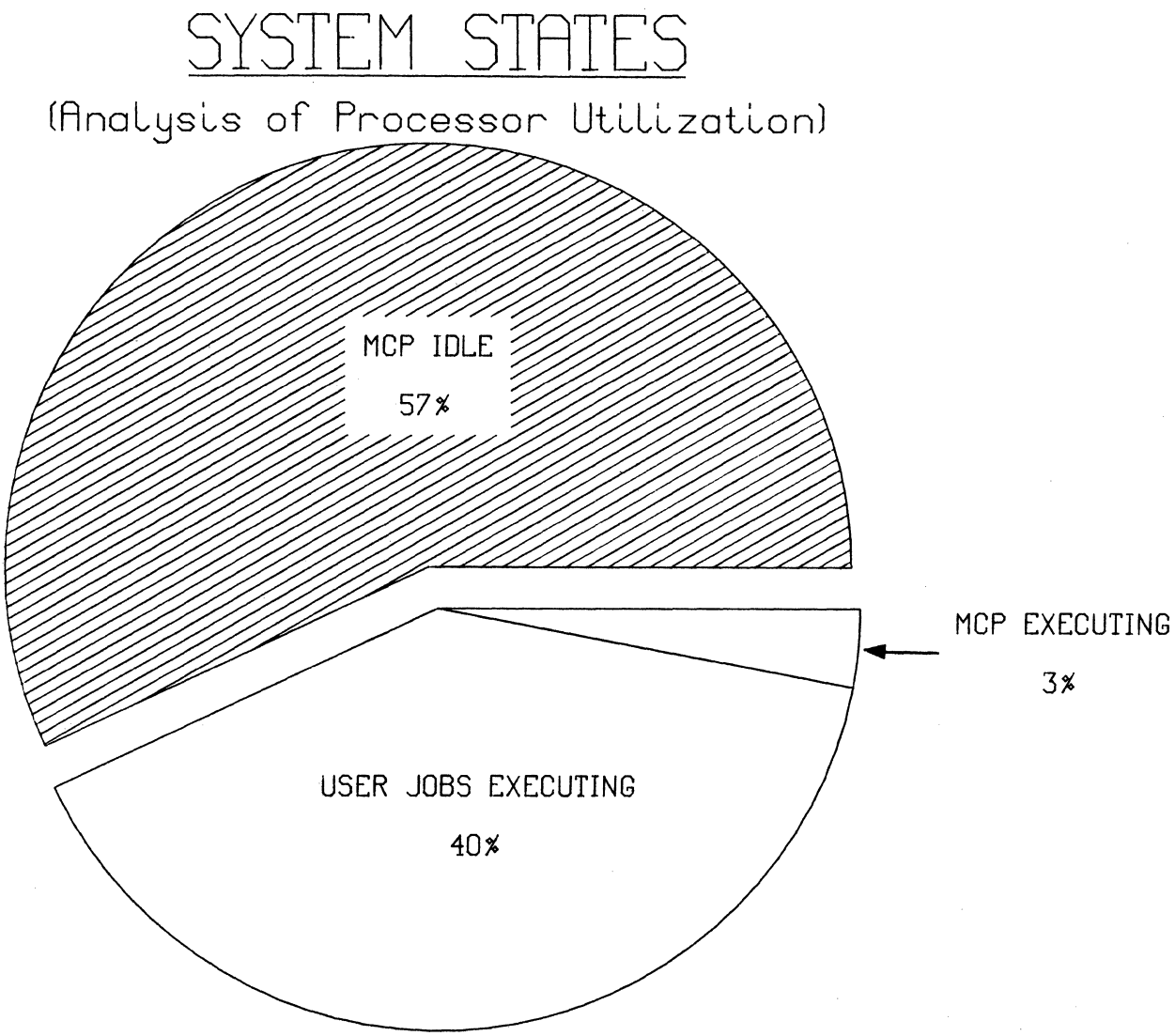


Figure 23-2. Comparison of MCP Idle Time to MCP and User Program Execution Time

STEP 1 - EXAMINE PROCESSOR UTILIZATION, FIND OUT THE MCP IDLE TIME

Examine your processor utilization to find out the amount of time the MCP spends waiting. If it idles more than 10 percent, a QWIK Disk optimization opportunity exists and you should proceed with the analysis. If not, the system is "processor bound", and QWIK Disk can do little to improve its total performance.

The MCP/USER graphs produced by the FLAME software package can help you determine the MCP idle time. If you don't have FLAME, a "soaker" program will perform a similar job. A soaker program is a priority 1 program that loops, continually counting, soaking up the unused processor time when no other program is executing and the MCP is idling.

The pie chart SYSTEM STATES illustrates a system in which the MCP idles (waits) 57 percent of the time. Another 40 percent is spent executing user jobs, while the MCP executes 3 percent of the time.

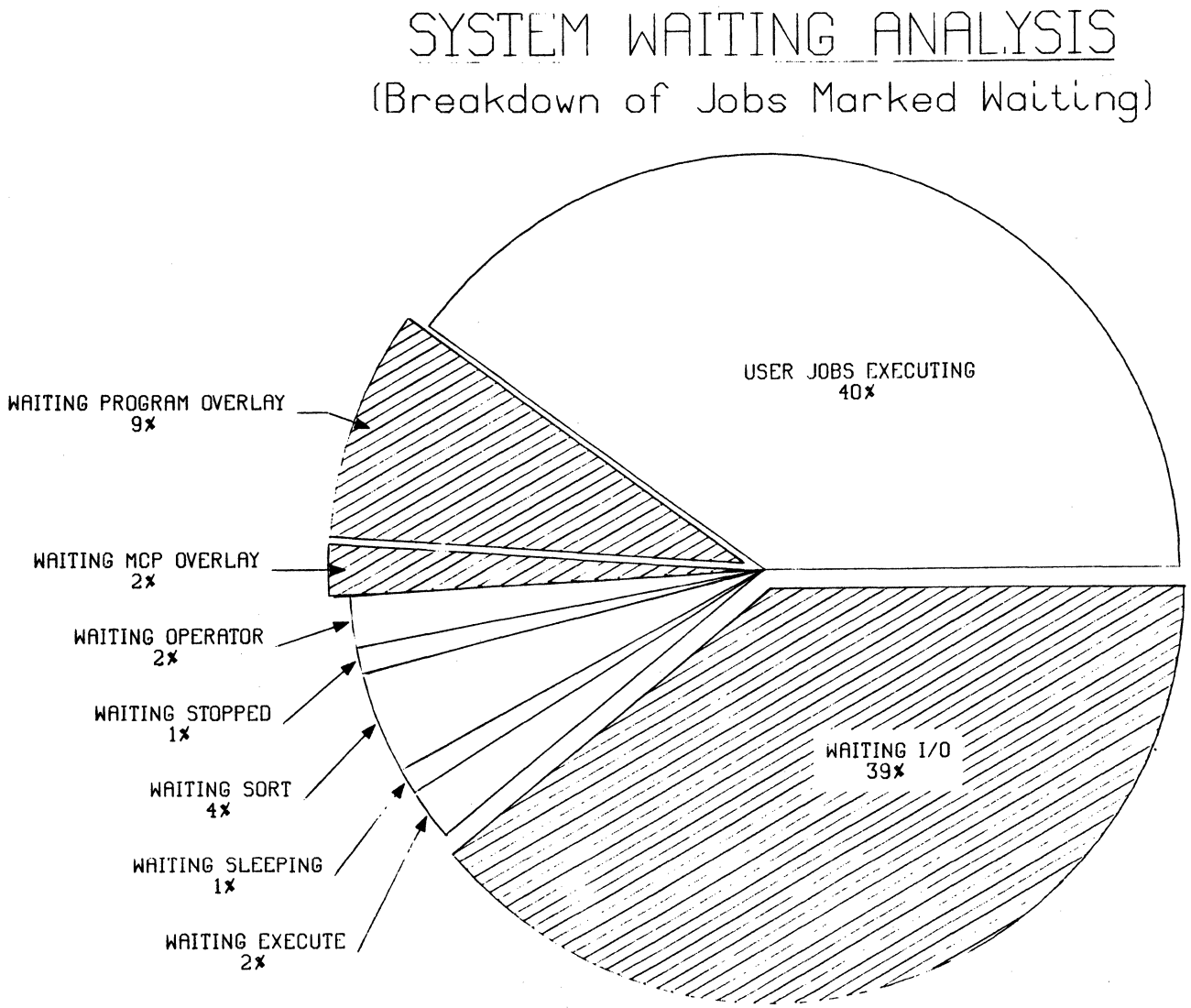


Figure 23-3. Factors that Contribute to MCP Idle Time

STEP 2 - HOW MUCH TIME IS SPENT WAITING MCP AND PROGRAM OVERLAYS

If your MCP spends more than 10 percent of its time waiting, determine what it spends its waiting time waiting for.

The pie chart "System Waiting Analysis" illustrates a number of factors that contribute to MCP waiting time. The bottlenecks QWIK Disk can help are listed below and are highlighted in the chart.

Time spent waiting overlays:

- MCP overlays
- program overlays

Time spent waiting I/O:

- I/Os to disk or pack
- I/Os to classes of files
- I/Os to individual files

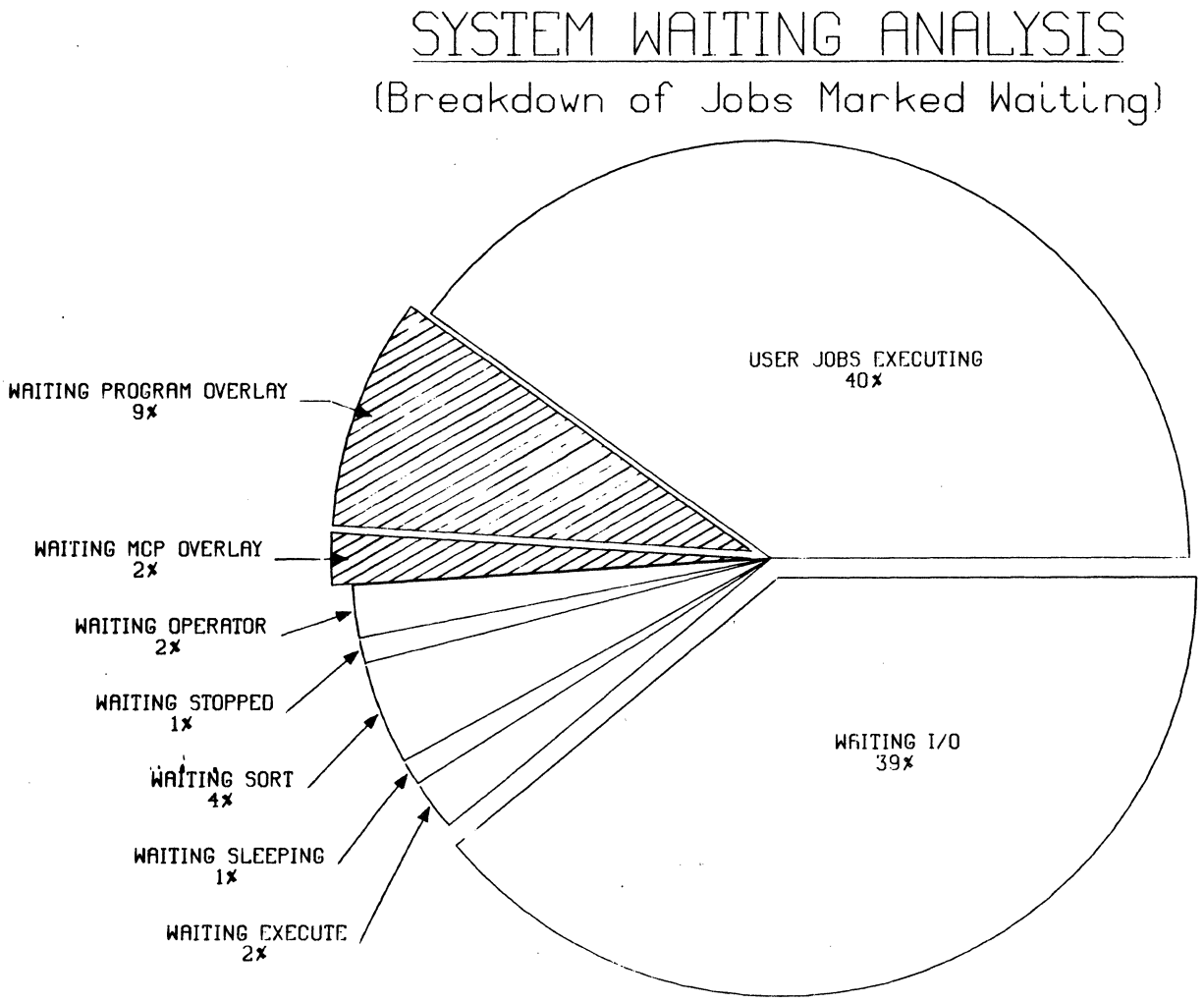


Figure 23-4. QWIK Disk I/O Performance For Programs Not Using QWIKPOOL

STEP 3 - MEMORY UTILIZATION, EXAMINE MCP AND PROGRAM OVERLAYS

Find out how much time is spent waiting for overlays.

MCP Overlays

Use the FLAME graph "Number of MCP Overlays versus Number of MCP Overlays from Disk" in the Memory Usage series of reports to get an idea of how much the system waits for MCP overlays. The QWKMEM option is the primary tool for speeding up MCP overlays. This option provides an area of main memory to be used exclusively for MCP overlays. The most recently called MCP overlay is stored in this area, so that when it is needed it can be retrieved from memory, eliminating disk I/O time. QWKMEM is discussed in the *V Series MCP/VS System Software Operations Guide, Volume 1: Installation.*

If there is much accessing of MCP overlays from disk even though the QWKMEM option is set, you can improve system performance by moving the MCP code and files to QWIK Disk. Keep in mind the QWIK Disk volatility issues. If more than 10 percent of waiting time is spent waiting for MCP overlays, this is a possible area for performance improvement.

Program Overlays

Now examine the amount of time programs spend waiting for overlays. The FLAME graphs "Percentage of Jobs Waiting versus Percentage of Jobs Waiting Program Overlays" and "Number of Program Overlays from Memory versus Number of Overlays" in the Memory Usage series of reports can be of help, as can the TABSII reports "QWIKPOOL Usage Summary" and "Average Overlay Call Hit Rate". If more than 10 percent of waiting time is spent waiting for program overlays, this is a possible area for performance improvement. The primary tactic is to increase the amount of memory allocated to the QWIKPOOL. The QWIKPOOL is an area of memory reserved for the most recently used program overlays, reducing the number of overlays that have to be read from disk. Otherwise, proceed with the following I/O analysis.

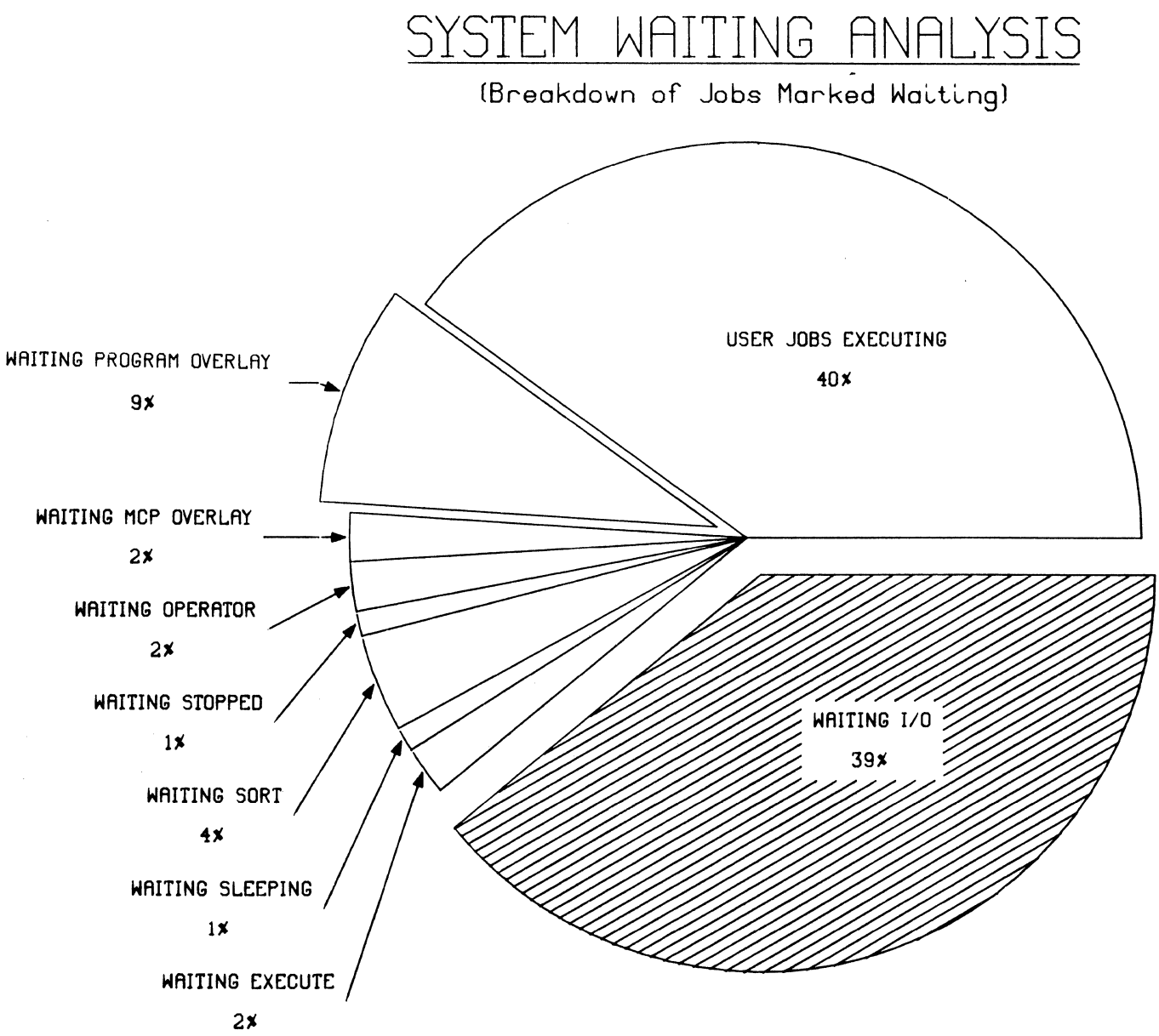


Figure 23-5. Find Out What Percentage of Time the MCP Spends Waiting I/O

STEP 4 - PERIPHERAL ACTIVITY, EXAMINE I/O WAIT TIME

Examine the RUNLOG I/O counts and the FLAME "Job Waiting" reports to find out how much time the MCP spends waiting I/O. If the MCP spends more than 10 percent of its waiting time waiting I/O, proceed with the following analysis. Find out how much time it spends waiting I/Os to disk or diskpack, MCP code files, various classes of files, and individual files. At this point you should be able to fit you system into one of the following categories:

Processor bound - jobs are executing at least 90 percent of the time. QWIK Disk can offer little performance improvement.

Memory bound - The system is waiting MCP or program overlays more than 10 percent of the time. QWIK Disk, QWKMEM and QWIKPOOL can improve overlay performance.

I/O bound - A high percentage of waiting time is spent waiting I/Os to peripheral devices and files. You can improve system performance by moving files with a high I/O volume to QWIK Disk.

I/O and memory bound - A high percentage of the system waiting time is spent waiting overlays and I/Os. First improve overlay performance by maximizing the QWKMEM and QWIKPOOL, then look into improving the time spend waiting I/Os. The steps you can use to isolate the files and peripheral devices are discussed in the following pages.

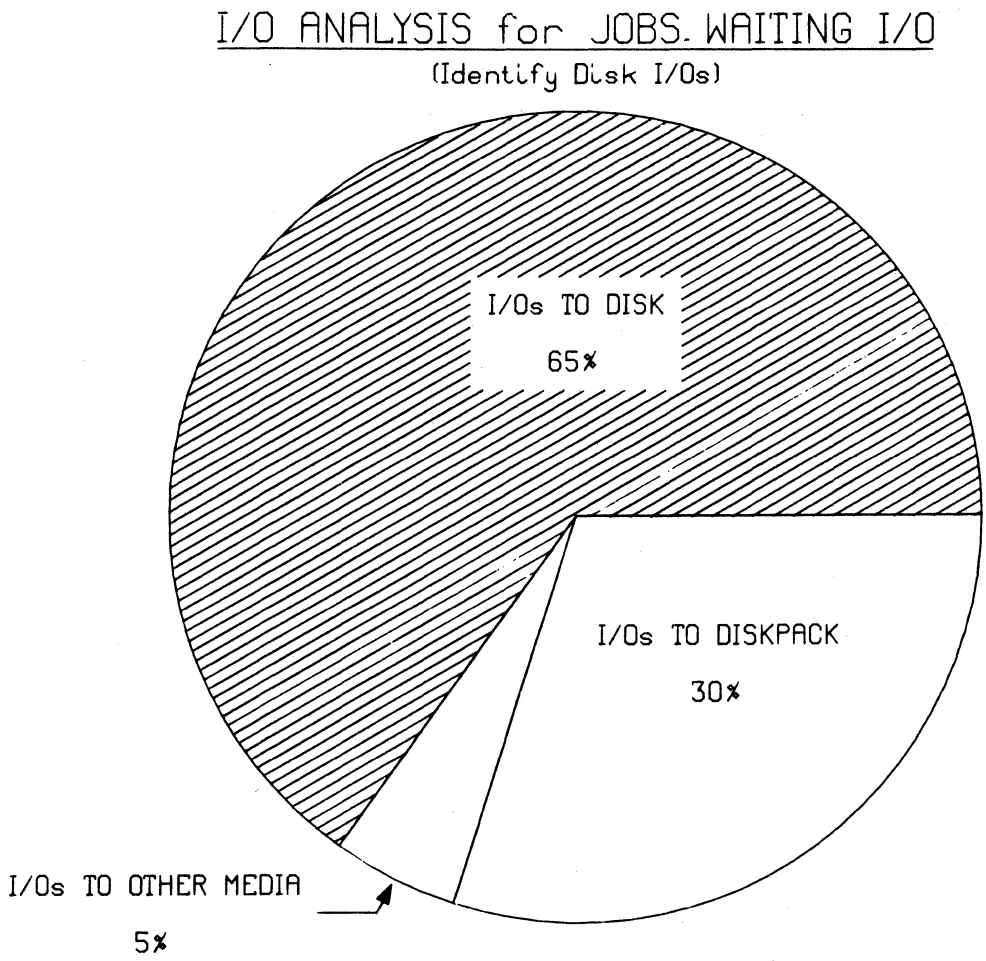


Figure 23-6. Analyze I/O traffic to Disk, Pack, and other Media

STEP 5 - I/O COUNTS TO DISK AND DISKPACK

If a majority of I/Os go to diskpack, your system can be characterized as a diskpack-based system. If the majority are going to disk, it is a disk-based system.

Diskpack Based - If your system is a diskpack based system, you can use QWIK Disk as a form of "cache" by moving files in and out as needed. This may require operational changes or program changes. Since these files are moved to QWIK Disk for temporary use, you must remember to copy them back to diskpack when you finish in order to update the original files. The analysis described in steps 6-9 can help you identify the types of files or the individual files with the highest I/O volume.

Disk Based - If your system is disk based system, proceed with the next analysis. The pie chart "I/O Analysis for Jobs Waiting I/O" illustrates a system that has 65 percent of its I/Os going to disk and 30 percent going to diskpack. The remaining I/Os go to other media. This system can be considered a disk-based system.

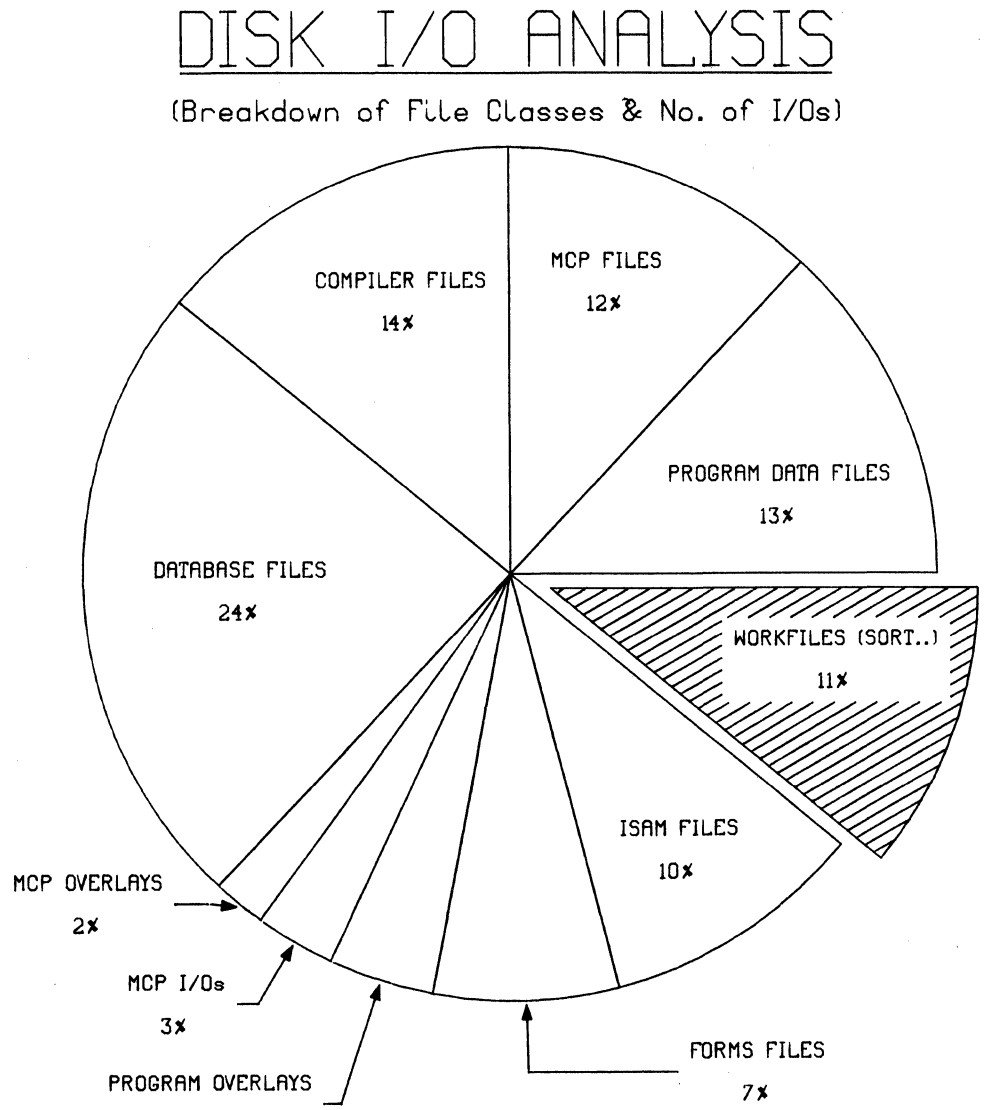


Figure 23-7. Disk I/O Analysis of a System Waiting I/O to 10 Different Types of Files

STEP 6 - PREPARE FOR GENERAL FILE I/O ANALYSIS

If excess MCP waiting time is spent waiting I/O, use FLAME and run log reports to find out what files or file types it waits for the most.

This file I/O analysis is broken down according to the following categories:

- high I/O to MCP files
- high I/O wait time on various classes of files including:
- work files
- data files
- data base files
- ISAM files
- high I/O wait time on individual files

STEP 7 - EXAMINE I/O VOLUME TO MCP FILES

Use FLAME graph 37, "MCP I/O", to determine if the MCP files have a high I/O volume. If they do, consider moving them to QWIK Disk. If not, proceed with the next analysis.

STEP 8 - EXAMINE I/O VOLUME TO CLASSES OF FILES

Use the run log to determine which files classes (such as database files) have the highest I/O volume. Make a list of the worst offenders and consider moving the smallest files with the highest I/O volume to QWIK Disk.

You can route certain classes of files to QWIK Disk either by declaring a work file subsystem at coldstart time, or by using such system options as WRKP. Further information about files selection appears in the subsection titled FILES. Work file subsystem information appears in the subsection titled PROGRAMMING CONSIDERATIONS.

STEP 9 - EXAMINE I/O VOLUME TO INDIVIDUAL FILES

Extrapolate specific file I/O counts and percentage of a job's total I/O wait time from the run log analysis. List the worst offenders and consider moving the smallest files with the highest I/O volume to QWIK Disk. Use the subsection File Selection Criteria to refine the list.

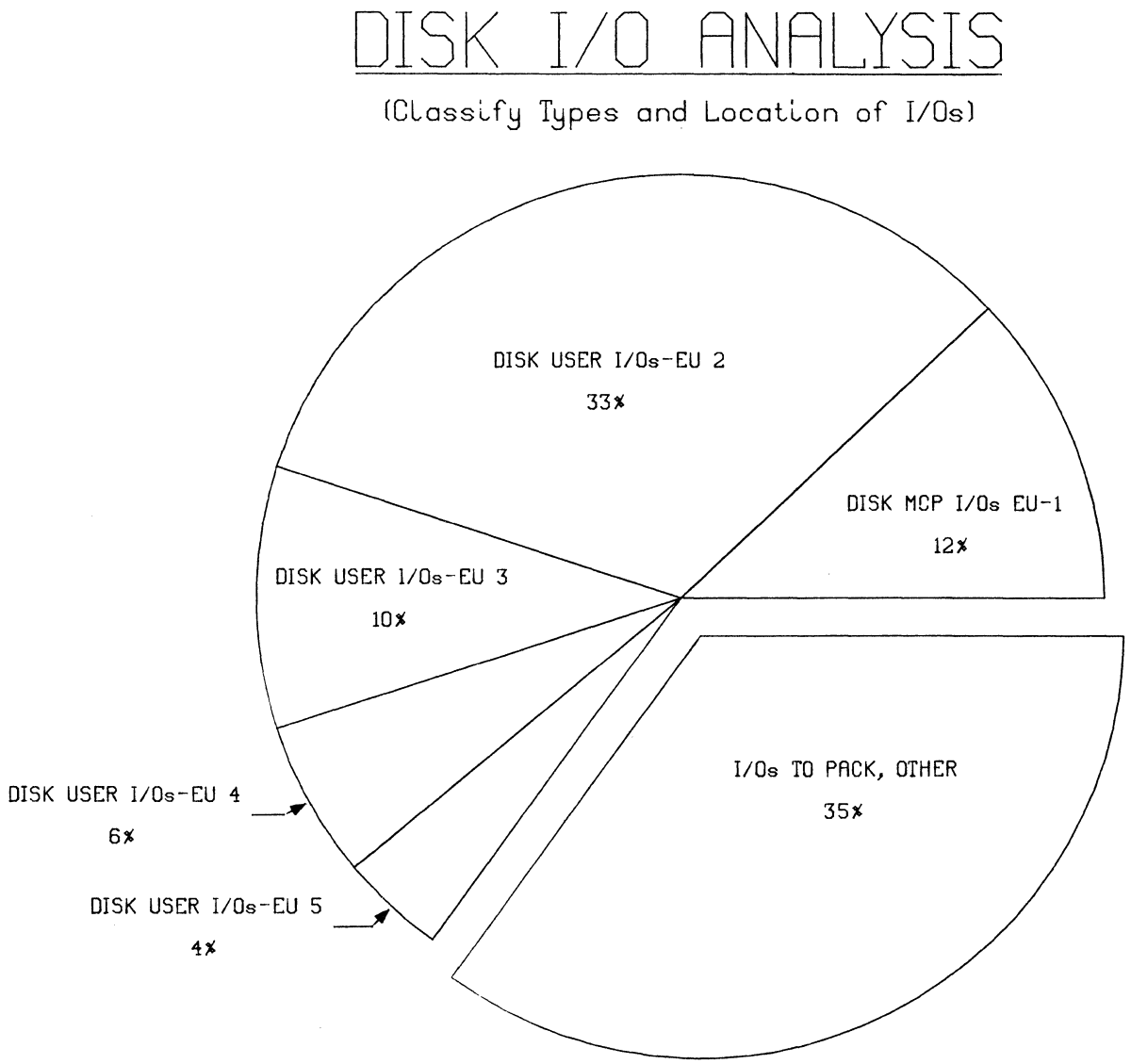


Figure 23-8. Underconfigured and Lopsided Peripheral Configurations

STEP 10 - CHANNEL/SUBSYSTEM/ID ANALYSIS

Underconfiguration and lopsided peripheral configurations can cause system bottlenecks.

If a disk ID or spindle is disproportionately busy, consider putting some of its files in QWIK Disk. Disk subsystem and ID analysis are available only in FLAME peripheral activity graphs.

The percentage of time a peripheral is in use should be divided evenly over the system's disk channels. If a disk channel shows a rate consistently over 40 percent, you may want to move some of its files to QWIK Disk. Because of its speedy I/O transfer rate, QWIK Disk use will rarely exceed 25 percent.

DYNAMIC PROFILE VERSUS STATIC PROFILE

After you follow the preceding analysis to characterize your system, you should be able to identify your system environment, its associated files, and I/O bottlenecks.

Since QWIK Disk is a volatile storage device (it loses its files if the power goes off), you should now determine whether the system would be best served by a constantly changing dynamic area, or by a static area where specific files can be loaded to and reside in QWIK Disk for the entire production cycle.

A static environment with files of permanent residency can still support some dynamic files. However, you should develop a backup procedure for all non-inquiry QWIK Disk files.

A static profile contains files with the following characteristics:

- Long file access span.
- Inquiry or limited update file
- ISAM
- Code files (for overlays)

A dynamic profile includes files with the following characteristics:

- Short file access span
- Temporary or output
- Work files

For dynamic files, consider the following question: Are the files in QWIK Disk recoverable by a user-coded restart?

SAMPLE ENVIRONMENTS

Following are some examples of how QWIK Disk might be used in different types of environments, including the development environment, the production environment, and the diskpack environment.

Sample Development Environment

In this sample environment, both dynamic steadily changing files, and static non-changing files are placed in QWIK Disk.

Dynamic compiles are frequently cycled through the QWIK Disk to achieve a high compile rate. Heavily accessed work files are also routed to QWIK Disk. QWIK Disk also holds static files, such as an inquiry-only file containing software reference sections.

Sample Production Environment

In the production environment, file access spans are generally long, file residency more permanent. A goal for this type of environment is to mitigate "bottleneck" structures. Critical issues include on-line response time and transactions per hour. This would be considered a "static" environment where some data update may occur.

Sample Diskpack Environment

In this environment, nearly all accessed files reside on 180-byte media. In this configuration, QWIK Disk serves as a type of diskpack cache, where heavily accessed diskpack files are copied to QWIK Disk for use. They are then copied back to diskpack to update the originals.

Because all QWIK Disk files are backed up on PACK, a QWIK Disk-resident MCP might be considered to facilitate MCP overlays and Directory I/Os. This option is not recommended unless the impact of directory loss is minimal. Keep in mind that if the disk directory is lost from QWIK Disk, a coldstart will be needed to recover.

Measurement Techniques

The following paragraphs outline the system commands and reports used to monitor system performance. This information can also help you decide which files to put on QWIK Disk. The reports tell you about the sizes of various files, as well as the number of program overlays and peripheral it uses.

The Keyboard Input Messages include:

- KA reports on file sizes and their location.
- KS reports all areas of disk usage by ID number
- OL reports on channel activity.
- WO reports on overlay activity in the QWIK pool.

The utility programs include:

- RLGOUT reports detail file sizes and I/O activity.
- FLAME reports help summarize QWIK Disk overall performance.

Using FLAME

FLAME is an effective tool for measuring certain performance areas. The system environment should be monitored both before and after QWIK Disk file placement.

The following graphs should be of particular interest.

Graph	Subject
30-37	Jobs in the mix marked waiting
11	Percentage of time user jobs executing
12	Percentage of time MCP executing and idling
21	MCP disk overlays
22	MCP main memory overlays
23	Number of program overlays from memory
42	MCP I/Os versus total I/Os to disk.
58	Percent of time channels are busy
57	Number of I/Os to channels

FILES

This subsection discusses criteria for selecting the files to be placed on QWIK Disk. It suggests certain files as possible candidates for placement on QWIK Disk. Finally, it discusses general procedures for loading and unloading files to and from QWIK Disk.

General File Selection Criteria

Generally speaking, QWIK Disk is ideal for files with a high I/O volume or that require a short response time or both. The performance improvement obtained by putting these files on QWIK Disk depends on the ratio of I/O time to processor time for the jobs that access the files.

A matter of equal importance is volatility. Always keep in mind that QWIK Disk is a volatile storage medium. If the power goes off, or if the system is coldstarted, all disk files, including those on QWIK Disk, will be lost.

The following table highlights the QWIK Disk file selection criteria; the paragraphs following the table discuss these criteria in greater detail.

	File Selection Criteria
File size	- is it less than QWIK Disk capacity?
	- certain areas heavily accessed?
	- can the file overflow to another ID?
File Access	- is it read-only or inquiry?
	- is it temporary or a work file?
	- is it an update file?
	- does it have codefile overlays?
File Volatility	- is restart feasible?
	- will data loss compromise integrity?
File Activity	- does it have a high I/O wait?
	- are large blocks being transferred?
	- does it need a high number of
	- I/Os per second?

The following paragraphs discuss the file selection criteria highlighted in the preceding table.

File Size

File Size - Files placed on QWIK Disk cannot be larger than the storage available.

File Areas Heavily Accessed - Sometimes it is possible to split a file and put the area used most on QWIK Disk, with the less-used areas on LAK or 5N disk.

File Overflow - When the storage capacity of QWIK Disk fills up, files can overflow to another disk ID. Overflow requires operator intervention on a area-by-area basis, using the OK and OU keyboard commands. This creates a situation in which there is high speed access to the part of the file in QWIK Disk, and low speed access to the part of the file that has overflowed to another disk ID.

File Access

Read-only Files - QWIK Disk offers high speed access to files that are accessed on a read-only basis.

Temporary and Work files - These files generally have a high I/O volume, and are used as logical extensions of main memory. Therefore, they benefit greatly from the access speeds and transfer rates provided by QWIK Disk. Consider, however, how volatile and critical the file is, since files on QWIK Disk are inherently more susceptible to loss than files on disk or diskpack. DMSII offers an alternative but recoverability is vital.

Update Files - The high-speed access to files offered by QWIK Disk can improve the response time in an on-line environment, in which files are updated with transactions coming in from a network. As with temporary and work files, the possibility of loss is a consideration, with the DMSII recovery capability an option.

Code File Overlay - The QWIKPOOL option should be used for program overlays. However, if specific programs have overlays that are too big for the QWIKPOOL, consider placing the entire code file on QWIK Disk. For additional information on the QWIKPOOL, refer to the discussions of the USE QWIK and LIMIT QWIKPOOL records in the *V Series MCP/VS System Software Operations Guide, Volume 1: Installation.*

File Volatility

Restart - Since QWIK Disk makes use of RAM memory, information stored in QWIK Disk can be lost as the result of a power failure, power down, or coldstart. If any files are lost, can they be restored or regenerated?

Coldstart Acceptable - If the MCP resides on QWIK Disk and the system loses power, it will be necessary to coldstart the system, reloading the MCP to QWIK Disk. This causes the loss of the master disk directory and, as a result, the loss of all data on disk.

File Activity

High I/O Wait Time - Files that are accessed randomly and that require a lot of I/Os generally have a high I/O wait time. These files are good candidates for QWIK Disk. I/O wait time for files that are accessed sequentially can usually be eliminated by using multiple buffers; but if this is not possible, these files can also be considered for placement on QWIK Disk.

High I/Os Per Second - If the run time of a job is directly proportional to the number of I/Os per second, you can reduce the run time by putting the file on QWIK Disk. Remember, put the smallest files with the highest I/O volume on QWIK Disk. QWIK Disk is also good for files that have large blocks to transfer, for small, frequently accessed files, and for files with high I/O traffic caused by a high transaction rate.

File Selection Suggestions

The following pages discusses the pros and cons of putting various classes of files on QWIK Disk. Included in the discussion are files used by the MCP, compilers, SORT, DMSII, ISAM, GEMCOS, and various kinds of form files.

For each class of file, an illustration suggests which files to put on QWIK Disk and which are best kept on other media.

COMPILER

File Sizing

Following are size estimates per 10,000 COBOL74 records as compiled by the COBOL compiler. In general, these figures average about 1 megabyte per 1,000 lines of code, or 500,000 bytes if the work files are assigned to disk. Fifteen megabytes of QWIK Disk can handle approximately 12,000 lines of COBOL code (total) if printer backup files are sent to hard disk. This assumes only one program compiling at a time.

Work files	5.6	Megabytes
Source	.9	
Code file	1.2	
Listing	1.6	
Total	9.3	Megabytes

How to Place Files in QWIK Disk

COBOL74, RPG and FORTRAN compiler work file placement is controlled by resetting the compiler option WRKP (RO WRKP). The QWIK Disk ID 40/0 must also be declared subsystem 8 in the coldstart parameters. This feature of MCP/VS causes all work files to first go to disk subsystem number 8, if there is one. Otherwise, they go to the default disk subsystem.

Compiler input files, source and COPY libraries can be put in QWIK Disk by using SYSTEM/COPY. See the *Operations*.

QWIK Disk Performance Expectations

Compilation rates vary depending upon listing options, database invocations, and work file locations. Also, the QWIK Disk storage capacity limits the number of compiles that can use QWIK Disk at the same time.

Work files constitute a large percentage (about 60) of the compiler I/O traffic. Merely placing work files on QWIK Disk should improve compilation rates by approximately 1.3 to 1.4 times.

900-Series compilers, such as COBOL, do few overlays. Placement of the compiler code files on QWIK Disk will produce little improvement on these systems, even without the QWIKPOOL option set.

COBOL copy libraries may cause numerous file OPENS and CLOSEs, as well as many I/Os. With heavy copy library usage, putting library files on QWIK DISK may improve compilation rates up to 3 times.

SORT

File Size

The work files for the SORT intrinsic, will use disk space about 2.25 to 2.6 times the actual size of the input file.

How to Place Files on QWIK Disk

SORT work files can be assigned to disk in the COBOL74 SORT SELECT clause or SORT statement in BPL. However, the QWIK Disk subsystem must be declared as the default subsystem to ensure work files will be allocated to the QWIK Disk.

QWIK Disk Performance Expectations

SORT: shows marginal improvement by placing its work files on QWIK Disk.

Placement of SORT input and output files will more significantly improve performance due to their higher I/O activity.

DMSII

File Size

DMSII file sizes may vary up to hundreds of megabytes, depending upon file attributes and file population. INDEXED SEQUENTIAL sets and subsets will generally be less than 1/4 the size of their associated datasets, and therefore will be better fits for QWIK Disk.

How to Place Files on QWIK Disk

The location of a DMSII file is declared in DASDL, in the file assignment specification part of the structure's physical attribute description. If the QWIK Disk ID or subsystem number is specified when the DASDL source is initially compiled, the structure will be created on QWIK Disk. If the system is to be coldstarted, the file must be moved between the backup medium and QWIK Disk, as does any disk file.

Alternately, the structure's physical attribute description can place the file on disk. The file can then be moved to QWIK Disk by using SYSTEM/COPY and specifying the ID or subsystem number of QWIK Disk.

If a DMSII file has previously been on another medium, its location can be changed to QWIK Disk by changing the file's physical attribute description in DASDL and recompiling the DASDL source with the $REORGANIZE or $UPDATE option set. If the $REORGANIZE option is used, the file will initially be moved by the Reorganize DBP, though after this it must be archived and restored to QWIK Disk like any other file. If the $UPDATE option is used, the file must be moved to QWIK Disk with the SYSTEM/COPY utility.

The DASDL syntax follows.

FAMILYNAME = DISK <disk assignment technique>

where :

<disk assignment technique>

BYID = <integer>

-or-

BYSUBSYSTEM = DSKn

(n = 1-8)

In the above syntax, <integer> is the ID number of QWIK Disk and n is the subsystem number of QWIK Disk.

QWIK Disk Performance Expectations

A DMSII database can consist of hundreds of structures, and only a fraction of them can realistically go on QWIK Disk. The smallest structures with the highest I/O volumes should be considered for QWIK Disk. Index sequential set and subset files consume much less space than data files, are accessed relatively often, and may therefore be good candidates.

QWIK Disk performance impact varies widely depending on the specific file placement. But performance factors of 1 to 4.7 can be achieved with placement of all or selected database files on QWIK Disk. The higher the number of database structure buffers, the less the impact of QWIK Disk (due to the reduction of physical I/Os). The DMSII DELETE and CREATE-STORE functions will usually generate the highest I/O wait time and will be most favorably impacted by QWIK Disk.

Recoverability of the database should be considered, as DMSII recovery or database rebuild using the audit trail will be necessary if the information in QWIK Disk is lost. In no case should the audit trail or the control file be put on QWIK Disk.

ISAM

File Sizing

COBOL74 and RPG programs offer an indexed sequential file organization (ISAM) and a relative file organization. ISAM "clusters" consist of data file and optionally a primary key file with up to 98 alternate key files.

ISAM and RELATIVE file sizes will vary widely depending on file attributes and file population. ISAM Indexed Sequential key files will generally be less than 1/4 of the data file size and are good QWIK Disk candidates.

How to Place Files on QWIK Disk

Output ISAM data files can be directed to QWIK Disk by the following syntax when the program is executed:

If QWIK Disk is the default subsystem:

```
? FILE <ISAM DATA filename> = <ISAM DATA filename> DSK
```

Alternately, this syntax can be used when the program is executed:

```
? FILE <ISAM DATA filename> = <ISAM DATA filename> QWK
```

QWK causes the MCP to assign the file to QWIK Disk at open output time.

For more information on the FILE command, refer to the *V Series MCP/VS System Software Operation Guide, Volume 2: Commands.*

Note that the associated key file cluster will also be directed to the location of the data file.

Existing ISAM file clusters that are already on disk can be copied onto QWIK Disk by using SYSTEM/COPY and specifying the QWIK Disk subsystem number as the destination.

Key files may be separated from large data files, and individually placed on QWIK Disk using this method.

Volatility should be considered if ISAM files are to be placed on QWIK Disk and a data recovery procedure should be established.

QWIK Disk Performance Expectations

ISAM functions will improve with data and key files located on the QWIK Disk media. Performance will vary depending on the number of buffers declared (default of 3).

High I/O wait ISAM functions like RANDOM WRITEs, DELETES, and RANDOM READs will show the biggest QWIK Disk improvement.

GEMCOS

File Sizing

The GEMCOS files with the highest activity are:

- Control File (GMCTL) Monitor Messages (GMSGS)
- Queue File (GMQ) Format File (GMFMT)
- Audit File (GMAUnn)

The Queue file size is determined by the QUEUEFILE and QUEUERECORDSIZE statements in the TCL source. Their size is usually less than 1 megabyte.

The Monitor Message file is read-only and generally contains fewer than 600 80-byte records.

The read-only Format File contains station message formats. The number of TCL formats will vary widely if the format option is used.

How to Place File on QWIK Disk

The GEMCOS TCL definition language may be used to direct various files to disk in the GLOBAL subsection:

CONTROLFILE = DISK _ DSK

FORMATFILE = DISK _ DSK

MONMSGFILE = DISK _ DSK

QUEUEFILE = DISK _ DSK

All GEMCOS files can be placed on QWIK Disk by SYSTEM/COPY by specifying the QWIK Disk subsystem number or ID number as the destination. All GEMCOS files can be file equated at run time.

QWIK Disk Performance Expectations

The MCS queue file will generally be used as an "overflow" for GEMCOS main memory input and station queuing. Although this should be an infrequent occurrence, QWIK Disk fast I/O turnaround can be exploited to get higher throughput in network peak period queuing.

Monitor Message and Format files are smaller read-only files which can easily be placed in QWIK Disk. Formats, in particular, may be accessed on every transaction. QWIK Disk can be used as a type of format "cache".

The GEMCOS Control file, although generally less than a megabyte, will sometimes create high I/O counts in an audited environment (SYNChronized recovery, CHECKPOINT recovery, WAITFORAUDIT). If performance bottlenecks become critical, consider moving the Control file to QWIK Disk. Note, however, that loss of the Control file information may compromise network recovery procedures. Archival recovery may be the only method to rebuild the control file.

CAUTION

Under no circumstances should the audit file be placed on QWIK Disk, as it is needed for MCS recovery.

Text deleted by PCN-001.

MCP Configuration Records for QWIK Disk

You must add two special MCP configuration records to your MCP configuration file in order to use QWIK Disk.

The first record reserves DLP channel 40 for QWIK Disk. If your system currently has a DLP assigned to channel 40, then the channel number for that DLP must be changed. Otherwise, that DLP will not be accessible when QWIK Disk is installed.

The second records tells the system how much storage QWIK Disk has.

The following system initialization record reserves channel 40 for QWIK Disk:

```
DLP 40 DSK
```

The following system initialization record tells the system how much storage QWIK Disk has and declares QWIK Disk as a non-shared subsystem. QWIK Disk must always be declared as a non-shared subsystem. For more information about using QWIK Disk on shared systems, see **Shared Systems** later in this subsection.

```
DISK 40/0 ID <nn> SUBSYSTEM <nn> 0 <available QWIK Disk sectors (or segments)>
```

You can find out the <available QWIK Disk sectors (or segments)> from the system's response to the commandTO ALL. If necessary, you can also convert the amount of memory allocated to QWIK Disk into disk sectors, by dividing the amount of QWIK Disk memory in digits (not KD or MD) by 200.

For more information about system initialization, see the *System Software Operations Guide, Volume 1: Installation.*

Putting the MCP on QWIK Disk

Because QWIK Disk loses all its information in the event of a power failure or power down, putting the MCP on QWIK Disk is not recommended. If the power fails, not only would it lose the MCP, but also the disk file directories, and all files on disk. Also, in environments that are memory-bound (to the exclusion of 500,000 bytes for QWKMEM), only marginal improvements can be expected over a 206 disk environment.

However, if you do decide to put the MCP on QWIK Disk, you need to add the following records to the system configuration file, in addition to those described above.

The following record helps you conserve space on QWIK Disk by directing the system log files to a disk media other than QWIK Disk. In this record, nn is a number other than that of QWIK Disk.

```
CONTROL LOGSUBSYS nn
```

The following record can be used to conserve additional space on QWIK Disk by directing the MCP dump file to a medium other than the QWIK Disk subsystem.

```
USE DUMP DISK SUBSYSTEM <nn>

USE DUMP PACK <family>

        or

USE DUMP TAPE
```

Text deleted by PCN-001.

Use QWKMEM and QWIKPOOL Options

Doing overlays out of QWKMEM is still faster that doing overlays out of QWIK Disk. Setting this option allows code files to be placed on a low speed disk and to achieve the same performance as if they were on QWIK Disk.

For information about the USE QWIK, LIMIT QWIKMEM, and LIMIT QWIKPOOL records and for more information about system initialization, see the *System Software Operations Guide, Volume 1.*

Shared Systems

Each processor in a shared system can have QWIK Disk. However, since QWIK Disk uses memory for storage, the files in QWIK Disk can be accessed only by the processor that is storing them. QWIK Disk does not have the capability of being exchanged. Therefore, QWIK Disk is declared a non-shared ID at coldstart. See the DISK initialization record in the subsubsection Coldstart above.

For consistency, we recommend that all QWIK Disk IDs in a shared environment be declared with the same subsystem number. This way you can specify QWIK Disk on all processors in the same manner. It also allows application programs that specify QWIK Disk to use QWIK Disk on any processor they run on.

Operations

QWIK Disk operation is treated much as any other disk ID. Files can be loaded and dumped in the same manner as any other disk. This subsection describes some operational procedures and enhancements that help optimize QWIK Disk performance.

Maintenance Processor Commands

The following maintenance processor commands are either new or were modified for QWIK Disk. How the system responds to these commands is discussed below.

- SO QWIKDISK [<nn>]
- RO QWIKDISK
- TO QWIKDISK
- LOAD MCP
- SO LIMIT
- SO MCP
- CLEAR MEM
- MEM ABS

Each time any of the first three commands are entered, the system displays the following.

QWIKDISK base = <nnnnnnnn>

QWIKDISK sectors = <nnnnnnnn>

QWIKDISK <status>

PHYSICAL MEMORY LIMIT SET TO <nnnnnnnn>

Caution on Maintenance Test Commands

The following maintenance test commands can destroy the files contained in QWIK Disk. If these commands are used, the system displays this warning:

*****WARNING*****
THIS COMMAND WILL DESTROY MEMORY/QWIKDISK CONTENTS

In order for the system to carry out the command, you must give it permission by appending the word "CLOBBER". If you use the word clobber the first time you use the command, the system will not display the warning, and it will carry out your command.

You are advised to copy the files in QWIK Disk to a safe medium before you use any of the following commands.

- CHAINT0 - T7 CLOBBER
- TEST CH T0 - T7 CLOBBER
- START MODTST <SPATH> CLOBBER
- CLEAR 0 CLOBBER
- CLOCK 0 CLOBBER
- TEST MEM CLOBBER
- DISPLAY T0 - T7 CLOBBER
- MSEL, MSHIFT, MCKEN, MGCKEN Commands

Powering On the System

Each time you power the system on, you must clear all the system's memory, including QWIK Disk memory. You must do this whether or not you are going to coldstart the system. See step 2 of subsection Pre Coldstart Procedure.

Haltload

A new function has been added to the LOAD MCP maintenance processor command. When you haltload a system that has QWIK Disk, the QWIK Disk memory area is rewritten and cleared of any transient single bit memory errors. No information in QWIK Disk is lost or changed, but this operation can make the haltload take a little more time to complete.

Putting Files on QWIK Disk

There are three steps involved in putting files on QWIK Disk:

- Before a program is started, selected files must be moved to QWIK Disk. They are moved either by using SYSTEM/COPY or they are programmatically created on QWIK Disk.
- The program executes, receiving performance benefits because it is accessing files from QWIK Disk.
- When the program finishes, files other than read-only files must be moved from QWIK Disk to other media for archiving or to update backup copies.

Files can be moved by the system operator using the SYSTEM/COPY utility, or a Work Flow Language (WFL) program can be written that will move the files to QWIK Disk, start the job, and move the files back to the original medium when the job finishes. Note that if temporary work files are created on QWIK Disk, they do not need to be moved from or to other media.

Using SYSTEM/COPY

SYSTEM/COPY can copy files to QWIK Disk. Just specify the QWIK Disk ID number or the subsystem number as the destination. See COPY in the *V Series MCP/VS System Software Operations Guide, Volume 2: System Commands* for the command syntax.

An example of the syntax is:

COPY MYFILE FROM DISK TO DISK (ID 1)

where QWIK Disk is ID 1

COPY MYFILE FROM DISK TO DISK (SUBSYSTEM 8)

where QWIK Disk is disk subsystem 8.

Loading and Unloading Files From QWIK Disk

Loading 15-megabytes of QWIK Disk from PE-tape may take from two to two and one half minutes.

Firmware

The firmware you load from the floppy PANDMV redirects I/Os to a "soft" channel/unit 40/0. Externally, this configuration looks like a 5N disk.

PROGRAMMING CONSIDERATIONS

This subsection discusses the programming considerations regarding QWIK Disk. It covers programmatic file creation on QWIK Disk, file equation to QWIK Disk, random and sequential I/Os, and code file overlays on QWIK Disk.

Creating Files

Files can be created for rapid access and existing file access can be optimized for specific time periods by moving files to QWIK Disk. Figure 2 shows relative benefits of QWIK Disk versus other peripheral devices.

Programmatic File Creation on QWIK Disk

Files can be programmatically created on QWIK Disk by using the file attributes or file declarations available with individual programming languages.

QWIK Disk as Default Disk

It is not recommended that QWIK Disk be used as default disk. QWIK Disk has a limited capacity that can fill up if all the files created on disk without a specific subsystem or ID number go to QWIK Disk. If QWIK Disk does fill up, the system looks for available space on non-default subsystems and it displays the following message:

NEED nnnn DSK SEGS FOR <file-id> <prog-id> = <mx number>

Then, to direct the QWIK DISK overflow to other subsystems, the operator must use the system commands OK or OU for each area that needs to be allocated.

Work File Subsystem Default

A feature of the MCP directs all compiler work files to the disk subsystem number 8, if there is one. If there is no disk subsystem number 8, then the work files will be placed as they have been on prior releases of the MCP.

The MCP option WRKP must be reset to take advantage of this option.

The purpose of this feature is to allow you to have compiler work files go automatically to QWIK Disk when QWIK Disk is declared as subsystem number 8. Since work files are temporary in nature and generally have a high I/O volume, they are ideal candidates for QWIK Disk. This feature is not restricted to compiler work files.

When the system displays the following message indicating that QWIK Disk has filled up, the operator can direct the overflow to another medium.

** NO USER DISK <job-specifier>

File Equation to QWIK Disk

To file equate files to QWIK Disk, use the following syntax:

FILE <file name> = <file name> QWK

This syntax causes the MCP at open (output) time to assign the file to QWIK Disk if there is any on the system. If there is no QWIK Disk, then the file goes to default disk. This change is also supported by WFL.

Hardware Type

QWK is the hardware type designation for QWIK Disk.

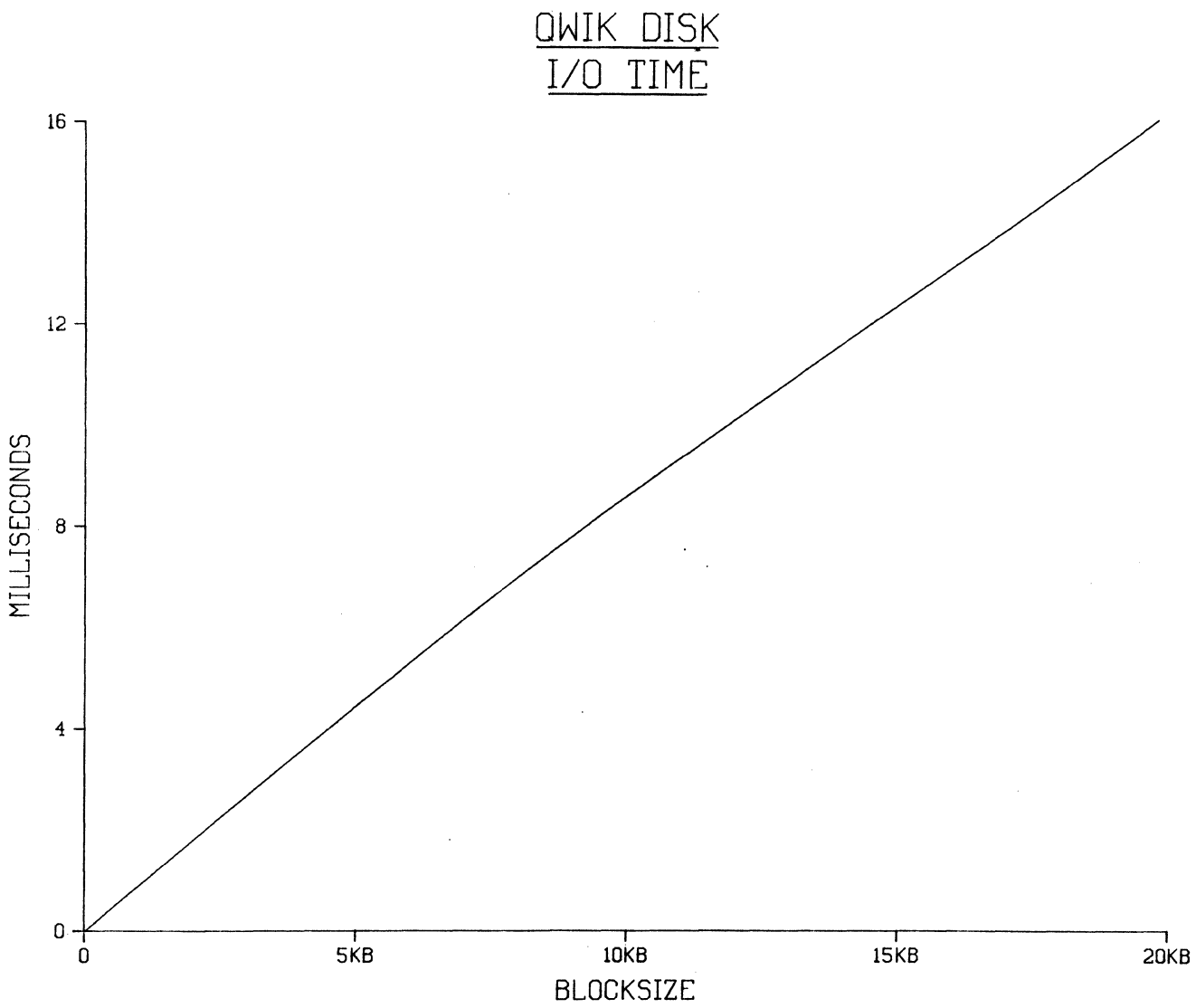


Figure 23-9. Blocksize and Time Factors for Completing I/Os to QWIK Disk.

Random and Sequential I/O

Random and sequential I/O access times are equivalent on QWIK Disk. The use of random or sequential I/O causes data to be stored in RAM memory. Therefore, the data transfer time is not dependent on seek time or latency positioning.

The amount of time it takes to complete a QWIK Disk I/O equals the amount of time it takes to transfer data. You can approximate the QWIK Disk I/O time by dividing the block length in bytes by the QWIK Disk data transfer rate of 1.5 megabytes per second.

Code File Overlays

The QWIKPOOL provides the best performance for code file overlays. However, QWIK Disk can be used in memory constrained environments or in a situations where many overlays exceed the size of the QWIKPOOL PAGE. If you route the oversize overlays to QWIK Disk, you can optimize the I/O wait time that would have otherwise been spent accessing the overlay on conventional disk. A QWIK Disk overlay of 9000 bytes costs about 6 milliseconds.

A Deviation from Standard Disk Operation

There is one other deviation from standard disk operation. Normally, when a write to disk does not fill out a sector, the point from the end of the data through to the end of the sector is filled in with zeros. QWIK Disk does not do this. For QWIK Disk, the write ends when the data runs out. The bytes remaining in that block stay set as they were set before the write.

ERROR CONDITIONS

This subsection describes the error conditions unique to QWIK Disk.

Result Descriptors

The three QWIK Disk hardware exceptions that can be reported are listed below. If the MCP encounters any of these errors, it reports it in the maintenance log.

Not ready (C000 8000)

> This indicates that an I/O operation tried to exceed the end of memory. The system thinks the memory limit is higher that it really is because at coldstart time the DSK card indicated that there were more segments available in QWIK Disk than there really are.

Data read error (C080 0021 0002 0000)

> This indicates that a read operation encountered a double bit error in QWIK Disk memory.

Memory error result (C400 0000 0000 0000)

> This indicates that a write operation encountered a double bit error in MCP memory.

Single Bit Errors

All single bit memory errors encountered in QWIK Disk are recorded in the maintenance log in the same way errors in MCP memory areas are recorded.

CAUTION

In the event of a power failure, or if the system is turned off, QWIK Disk loses all the information it is storing. Since the V- Series QWIK Disk does not provide battery back up, you may want to use an Uninterruptible Power System (UPS) as back up. In the case of a power failure, the power from the UPS could be used to copy the QWIK Disk files to tape, diskpack, or hard disk.

SECTION 24
COMMUNICATIONS PROCESSOR FIRMWARE LOADING PROGRAM (CPLOAD)

CPLOAD is a utility program provided as part of the V Series Communication System (VCS). CPLOAD loads firmware and files to communications processors (CPs) that are connected to the host system through VCS. Currently, VCS supports two types of communication processors: a TELCOM DLP and a CP 3680.

- For TELCOM DLPs, CPLOAD provides the same functions as the utility LDHOST in a non-VCS environment
- For CP 3680s, CPLOAD provides the functions of the CP 3680-specific programs FDPLOD and FDPRES. These functions include firmware loading and file transfer.

INITIATING CPLOAD

CPLOAD is initiated after:

- An operator enters an LH command using the CP <CP number> or the NCP <NCP number> syntax option.
- A label containing the value "FE*IBL" is detected in the input received from a CP 3680
- VCS software requests the operating system to initiate CPLOAD

LH Command

CPLOAD can be initiated by VCS-dependent options of the LH command. CPLOAD performs specific actions in response to other syntax options of the LH command.

WARM

When the LH option WARM is entered, CPLOAD compares the versions of the firmware files residing on the CP and on the host system. CPLOAD loads only those firmware files that are present in *different* versions on the CP and the host system.

When the WARM option is not entered, all firmware files are unconditionally loaded to the CP.

DUMP

CPLOAD responds differently to the LH option DUMP depending on the type of CP; TELCOM DLP or CP 3680.

- For TELCOM DLPs, CPLOAD dumps TELCOM memory into a file on the host system *before* loading firmware.
- For CP 3680s, CPLOAD transfers the system parameter file of the CP 3680 into a file on the host system before loading firmware.

In either case, the host system file is named "CxxxyP". Here, xxx reprents the first three digits of the CP number and y represents the hardwired processor number of the host system where

CPLOAD is executing. For example, if the LH command LH CP 14 DUMP is entered on system number 2, CPLOAD to creates a dump file named C0142P.

Dump files from TELCOM DLPs can be analyzed with the utility program DC1ANL. System parameter files from CP 3680s can be analyzed using the utility program VCSPRN.

FE*IBL Label (CP 3680 only)

When a label with the value "FE*IBL" is detected in the input from a CP 3680, CPLOAD performs the functions of the CP 3680-specific program FDPLOD. CPLOAD loads only a code file into the CP's memory. This is sometimes referred to as "host booting the CP".

VCS Initiation

VCS software can respond to a wide variety of conditions in a data communications network. In several situations, VCS instructs the operating system to execute CPLOAD, passing specific instructions that CPLOAD can interpret. The VCS software may request that certain specific files be unconditionally loaded to the CP. When CPLOAD receives such a request, it determines if any additional files must be loaded to the CP and then loads all required files.

ERROR MESSAGES

CP3680 SYSTEM FILE UPLOAD ABORTED

An unrecoverable error has occured during a dump of a CP 3680. Any information transferred to the host system before the dump was aborted is retained.

ERROR FOR FEP REQUEST: FC = xx, SFT = xx, EC = xx, P1 = xx

This message is returned from a CP 3680 when the CP 3680 detects an error during a function request. The fields displayed represent values returned by the CP 3680, including, the function code, the sub-function, the error code and parameter 1.

INAPPROPRIATE REQUEST FOR TELCOM

This message is returned from a TELCOM DLP in response to a load request or other function.

INCORRECT CP RECORD: SOUGHT # xxxxxx, FOUND # xxxxxx

Certain data in the VCS initialization file is incorrect. This message is usually caused when the Network Controller (NC) has failed during an update, before the update was complete. The SOUGHT field shows the number of the CP that CPLOAD was looking for, the FOUND field shows the number of the CP that CPLOAD was able to find.

INCORRECT INDEX RECORD: SOUGHT # xxxxxx, FOUND # xxxxxx

Certain data in the VCS initialization file is incorrect. This message is usually caused when the Network Controller (NC) has failed during an update, before the update was complete. The SOUGHT field shows the number of the CP that CPLOAD was able to find.

INIT FILE CONTAINS NO RECORDS

CPLOAD is unable to read any information from the VCS initialization file. This message is encountered very infrequently and indicates that the operating system passed incorrect information about the location of the VCS initialization file to CPLOAD. CPLOAD may be attempting to read a non-VCS file.

INIT FILE NOT FOUND

The VCS initialization file is not present. This message is encountered very infrequently and indicates that the operating system passed incorrect information about the location of the VCS initialization file to CPLOAD.

INIT REC MISMATCH: SOUGHT xxxxxxxxxxxx, FOUND xxxxxxxxxxxx

Certain data in the VCS initialization file is incorrect. This message is usually caused when the Network Controller (NC) has failed during an update, before the update was complete. The SOUGHT field shows the type of VCS entity that CPLOAD was searching for, the FOUND field shows the type of entity that CPLOAD was able to find.

INSUFFICIENT TELCOM MEMORY FOR REQUESTED FUNCTION

This message is returned from a TELCOM DLP in response to a load request or other function.

INVALID INIT FILE FORMAT: xxxxx BYTES, BLOCKED xxx

The VCS initialization file is in the wrong format. This message is encountered very infrequently and indicates that the operating system passed incorrect information about the location of the VCS initialization file to CPLOAD.

INVALID INIT FILE KEY: xxxxxxxx, TYPE xxxxxxxxxxxx

Certain data in the VCS initialization file is incorrect. This message is usually caused when the Network Controller (NC) has failed during an update, before the update was complete. The KEY field shows the key value CPLOAD was searching for, the TYPE field shows the type of VCS initialization record that CPLOAD was looking for.

NETWORK FILE ERROR: INVALID HEADER: ERROR DIGIT INDEX = xxxx

	VER	FLGS	NIO	CP	NCP	EC	ERRDIG	DATASZ	CPHDRSZ
HEADER =	x	x	xx	xxxx	xxxx	xx	xxxxxx	xxxx	xxxx

This is a specific Network File error that occurs often enough to be included here. The error digit index gives the position of the incorrect digit in the Network File direct I/O header. Possible causes for this error include invalid data passed to CPLOAD by the operating system, invalid data in the VCS initialization file or inconsistencies within CPLOAD.

NETWORK FILE ERROR: <error text>

Some error conditions relating to the Network File may occur during the execution of CPLOAD. These conditions occurs very infrequently, but when they happen, the error message is displayed by CPLOAD preceded by NETWORK FILE ERROR.

NO HOST CONNECTED TO THIS CP CLAIMS TO BE MASTER

CPLOAD cannot find a master host system in the CP 3680 initialization file. This message is encountered very infrequently.

TELCOM DUMP ABORTED

An unrecoverable error has occured during a dump of a TELCOM DLP. Any information transferred to the host system before the dump was aborted is retained.

TELCOM FUNCTION NOT DONE, NO REASON GIVEN FOR FAILURE

This message is returned from a TELCOM DLP in response to a load request or other function. The TELCOM has not returned any data to CPLOAD.

TELCOM LINE AND STATION MUST BE NOT READY FOR THIS FUNCTION.

This message is returned from a TELCOM DLP in response to a load request or other function.

TELCOM TABLE NOT DELETED, OUTPUT QUEUE IS NOT EMPTY

This message is returned from a TELCOM DLP in response to a load request or other function.

TELCOM TABLE NOT DELETED, OUTPUT QUEUE IS NOT EMPTY

This message is returned from a TELCOM DLP in response to a load request or other function.

TIMESTAMP OF ZERO FOUND IN HOST FILES

A timestamp with a value of zero exists in one or more of the host system's VCS files. This message is encountered very infrequently. Possible causes include an incorrect file name in the CP or HOST records of the VCS initialization file, or a problem with the Network Controller (NC).

UNRECOGNIZED INITIALIZATION PARAMETERS VERSION

A mismatch has occurred between the versions of CPLOAD and the operating system. This is caused by differing release levels of the operating system and CPLOAD.

UNRECOGNIZED RBI RECEIVED FROM TELCOM, RBI = xx

This message is returned from a TELCOM DLP in response to a load request or other function.

xxxxxxxxxxxx: EXPECTED TO FIND xxxxxx, ACTUALLY FOUND xxxxxx

Certain data in the VCS initialization file is incorrect. This message is usually caused when the Network Controller (NC) has failed during an update, before the update was complete. The first field in the message shows the VCS entity type (for example STATION), the EXPECTED TO FIND field shows the expected number of entities, the ACTUALLY FOUND field shows the actual count.

This message indicates an error condition, but CPLOAD continues loading firmware files.

ZERO OR NON-MATCHING TIMESTAMPS RETURNED FROM CP

The timestamps of the firmware files on the CP do not match the timestamps of the firmware files on the host, even after CPLOAD has loaded new firmware files. This message is encountered very infrequently. Possible causes include a CP in a state where file loading is invalid, or errors detected by the CP during the loading of files. When files are loaded to a CP 3680, CPLOAD does not detect any errors, but the operating system does display error messages.

SECTION 25
DEBUG FACILITY

The MCP Debug Facility is an interactive, menu-driven tool for debugging the MCP and user programs at the machine language level. You can run up to ten Debug sessions at one time.

You initiate a Debug session by entering one of two commands at an ODT. The ODT then becomes an interactive terminal for that Debug session. The Debug facility allows you to switch back and forth between any of the active Debug sessions, and between a Debug session and the normal operations of the ODT.

INITIATING A DEBUG SESSION

There are two commands that initiate the Debug Facility:

- The DEBUG command initiates a program and attaches the program to a Debug session.
- The Interactive Debug (ID) command creates a Debug session for a program that is a task in the active mix.

After you have entered either of these commands, the ODT becomes an interactive terminal for a Debug session. The terminal is reset and internally reprogrammed for the Forms Mode. A set of user interface menus will appear.

Following the Debug session, the terminal will be reinstated to its original state, and all screens will be cleared.

MCP Initialization

In order to use the Debug Facility commands, you must include either the CONTROL DEBUG MCP configuration record or the CONTROL DEBUG USER configuration record in the MCP configuration file at system initialization.

The CONTROL DEBUG MCP configuration record allows you to use the interactive DEBUG facility with the MCP or with user programs. The CONTROL DEBUG USER configuration record allows you to use the interactive DEBUG facility only with user programs.

For a complete description of MCP configuration records, refer to the *V Series MCP/VS System Software Operations Guide, Volume 1: Installation.*

UTILITY COMMANDS

There are two utility commands which are useful in running a number of Debug sessions concurrently:

- The Enter Debug (ED) command returns you to any of ten Debug sessions from the ODT.
- The Query Debug (QD) command displays the number of active Debug sessions on the system and their respective session numbers.

These two utility commands are useful because you can leave the Debug Facility and return to normal ODT operations at any time by using the ODT action on the Main Menu. (The ODT action and the Main Menu are described later in this section.)

THE DEBUG COMMAND

The DEBUG command starts a program and simultaneously attaches the ODT to a Debug session for that program. This command is entered only through an ODT. You can run up to ten Debug sessions concurrently.

The syntax for the DEBUG command is as follows:

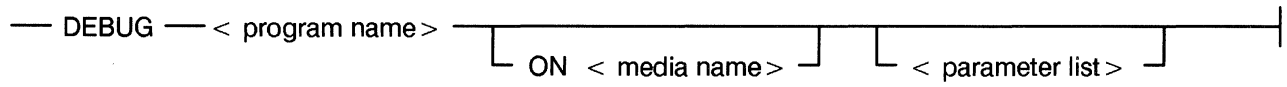


Figure 25-1. DEBUG Command Syntax

<program name>

is the name of the program that will be attached to an interactive Debug session. The program must contain executable object code. The program must be stored on the disk or pack specified as <media name>.

<media name>

refers to the name of the storage device where the desired program is stored. If you do not specify the <media name>, the program must reside on disk. The <media name> can be one of the following:

DISK

looks for the desired program on 100-byte media only. If the program is not found, an error message is displayed on the ODT.

PACK

looks for the desired program on all of the system's unrestricted packs. If the program is not found, an error message is displayed on the ODT. The program cannot be found unless it is stored in 180-byte code file format.

To ensure that the program is stored in the correct format, copy it onto the pack using the PCOPY utility or compile the program using the <code file media id> option of the COMPILE (Compile Program) command. Refer to the *V Series System Software Operations Guide Volume 2: System Commands.*

<pack family name>

looks for the desired program on the packs of the entered family name. If the program is not found, an error message is displayed on the ODT.

If <media name> is not entered, the operating system will search all 100-byte disks for the desired program. If the program is not stored on disk, the operating system will continue searching for the program on all unrestricted packs. If a CONTROL CODEPACK <family-id> MCP configuration record was included in the MCP configuration file at system initialization, that family-id will be searched in place of unrestricted packs (refer to the *V Series MCP/VS System Software Operations Guide, Volume 1: Installation* for a complete description of MCP configuration records).

If the program is not found on disk or on unrestricted pack, an error message will be displayed on the ODT.

<parameter list>

can include up to three parameters to be inserted into a program when the program is started.

These parameters allow you to enter boolean, string, and integer values that can be used by the program when the program is being executed.

The syntax for <parameter list> is shown in Figure 25-2.

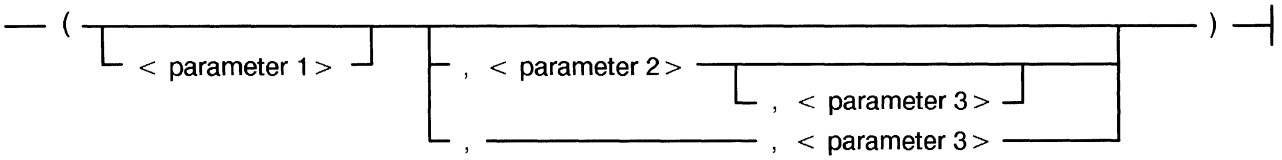


Figure 25-2. Syntax for Parameter List

The rules for <parameter list> resemble those for parameters in the Work Flow Language (WFL) RUN command. Refer to the *B 2000/B 3000/B 4000/V Series WFL Compiler Programming Reference Manual.*

Integer parameters:

- contain unsigned numeric data up to eight digits in length (8 UN).
- have values ranging from 0-F.
- are right-justified and zero-filled in the high-order digits if less than the maximum length.
- will cause a syntax error if greater than the maximum length.

String parameters:

- contain alphanumeric data up to six characters in length (6 UA).
- must be within quotes. To pass a quote as a parameter, a double quote must appear in the parameter string; for example: DEBUG MYPROG (,"TRY""XX").
- may be composed of any EBCDIC characters.
- are left-justified and blank-filled in the low-order bytes if less than the maximum length.
- will be truncated in the low-order bytes if greater than the maximum length.

<parameter 1>

Can be an integer up to eight digits in length. This parameter sets switches at program address 0; for example: DEBUG MYPROG (10100000).

<parameter 2>
<parameter 3>

Can be eight-digit integers or six-character strings.

<parameter 2> is inserted into the code file at base-relative address 8; <parameter 3> at base-relative address 20; for example: DEBUG MYPROG (,12345678) or DEBUG MYPROG (101,"TOTEST").

When either <parameter 2> or <parameter 3> are specified, you must have either a value or a comma in the preceding parameters; for example: DEBUG MYPROG (,,"TAPE").

A null entry is allowed. For example, DEBUG MYPROG (,0,"ABC") does not change the switches and inserts 00000000 at address 8 and the string ABC at address 20.

Examples

DEBUG PAYRL1

looks for a program named PAYRL1 on the disk named CODEPACK, or on unrestricted pack, starts the program and attaches it to an interactive Debug session.

DEBUG UPDATE ON PACK

looks for a program named UPDATE on all unrestricted packs, starts the program and attaches it to an interactive Debug session.

DEBUG STAT01 ON ACCTS ; FILE OUTPUT = LINEPR/ST1985

looks for a program named STAT01 on the pack family named ACCTS, starts the program and directs the file named OUTPUT to a specific printer.

THE INTERACTIVE DEBUG COMMAND (ID)

This command is used to debug a task in the active mix. A Debug session is invoked at the ODT from which you issue the ID command. The task corresponding to the task number is then attached to the Debug session.

You use this command to debug a task that is in an indeterminate state of execution, such as in a deadlock, an infinite loop, or during the execution of an Independent Runner.

The syntax for the ID command is as follows:

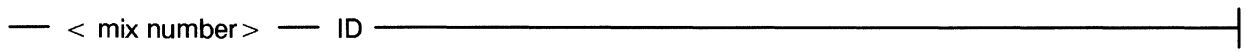


Figure 25-3. ID Command Syntax

<mix no>

refers to the mix number of an active job in the mix.

Example

55 ID

invokes the Debug Facility for the program associated with mix number 055.

THE ENTER DEBUG COMMAND (ED)

The ED command allows you to go back to a specified Debug session from the ODT.

The syntax for the ED command is as follows:

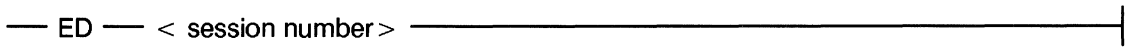


Figure 25-4. ED Command Syntax

You can go back to any Debug session from the ODT, as long as the Debug session was initiated from that ODT. The session number must follow the keyword ED because it identifies an Independent Runner that controls the session (as opposed to a normal Task Number).

The four-digit session number is prominently displayed within each session. The QD command (described later in this section) allows you to view the session numbers of all active Debug sessions.

Example

ED 1387

causes you to re-enter Debug session number 1387.

THE QUERY DEBUG COMMAND (QD)

The QD command identifies the Debug sessions present on the system and displays the four-digit session number of each. You must enter this command from the ODT.

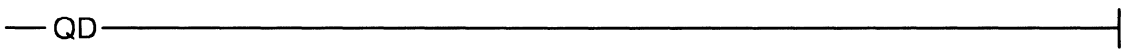


Figure 25-5. QD Command Syntax

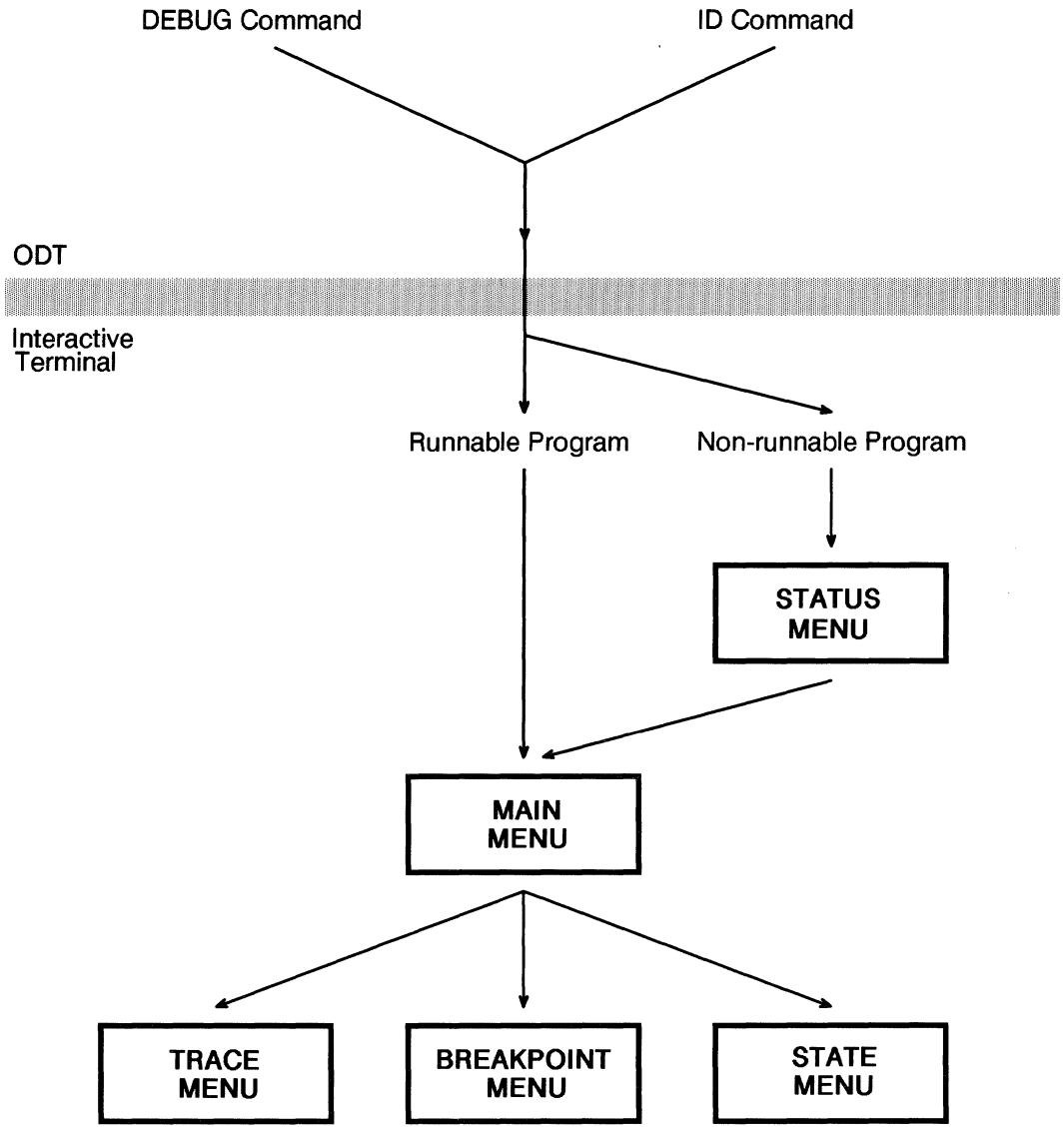


FIgure 25-6. User Interface Menus

THE USER INTERFACE MENUS

The Debug Facility contains several menus as well as an on-line help facility. These menus are:

- Main Menu
- Status Menu
- Trace Menu
- Breakpoint Menu
- State Menu

Figure 25-6 illustrates the series of user interface menus.

The Main Menu appears after a DEBUG or ID command when the task is in a runnable state. If the task is non-runnable, the Status Menu appears.

The Trace Menu allows you to trace and record the execution of the task up to a specified point.

The Breakpoint Menu allows you to set breakpoints in the instruction stream where you want execution of the program to stop.

The State Menu gives you access to memory for the purpose of examining and/or altering memory contents.

The State of a Debugged Task

All of the menus display the state of the debugged task on the second line (refer to Figure 25-7). This information includes the following:

TASK The task number assigned to the task.

MODE The Debug mode (USR, MCP, SYS, or REL).
Refer to the description of Mode later in this section.

STATE The internal state of the task:

DEBUG	presently debugging
STOP	stopped
GO ON	running
BRK-PNT	Breakpoint Menu invoked
HCL-BPT	Hypercall/BCT breakpointed
ADDR-BPT	address breakpointed
OP-BPT	Opcode breakpointed
OVF-BPT	overflow breakpointed
BR-BPT	taken branch breakpointed

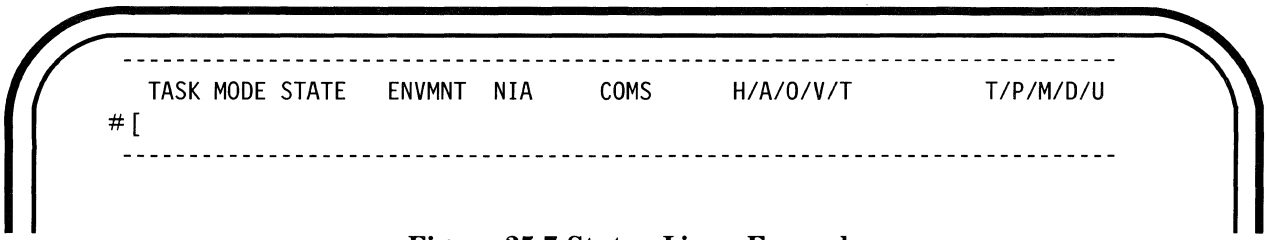


Figure 25-7 Status Line - Example

ENVRMT The task's current environment. This is a six-digit field; the last three digits are the module number.

NIA The next instruction address to be executed.

COMS The comparison toggles. Refer to the description of this field under The State Menu later in this section.

H/A/O/V/T An asterisk indicates which breakpoint has been used to stop the program:

H —Hyper Call / BCT
A —address
O —opcode
V —overflow
T —taken branch

Refer to the description of the Breakpoint Menu later in this section.

T/P/M/D/U An asterisk indicates which trace action has been set:

T —trace
P —path
M —monitor
D —module
U —user

Refer to the description of the Trace Menu later in this section.

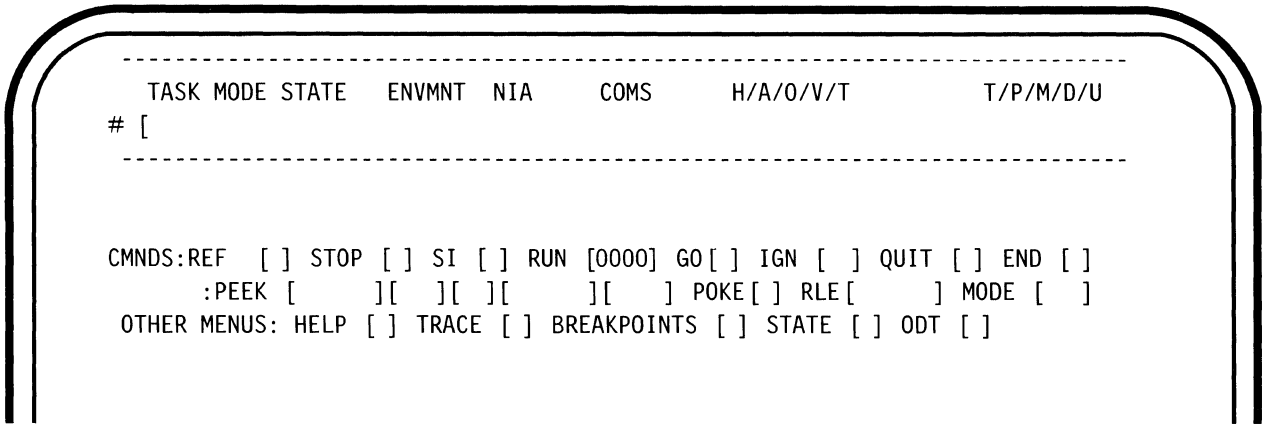

TASK MODE STATE ENVMNT NIA COMS H/A/O/V/T T/P/M/D/U
[
CMNDS:REF [ ] STOP [ ] SI [ ] RUN [0000] GO[ ] IGN [ ] QUIT [ ] END [ ]
:PEEK [ ][ ][ ][ ] POKE[ ] RLE[ ] MODE [ ]
OTHER MENUS: HELP [ ] TRACE [ ] BREAKPOINTS [ ] STATE [ ] ODT [ ]

Figure 25-8. Main Menu

The Main Menu

The Main Menu (refer to Figure 25-8) appears when you invoke a DEBUG or ID command (described earlier in this section), but only when the task to be debugged is in a runnable state.

If the task is initially in a waiting state, such as when it is waiting on a resource, the Status Menu appears (refer to Figure 25-9). The Main Menu appears when the task becomes runnable.

From the Main Menu, you can access the Trace, Breakpoint and State menus for the purpose of executing code in an observable, controlled manner.

Main Menu Actions

The following actions can be invoked on the Main Menu.

REF — Refreshes the screen by redisplaying the current menu.

STOP — Causes the task to stop the execution of instructions.

SI — Allows single instruction execution.

RUN — Allows execution of a specified number of instructions before stopping again.

The syntax for the RUN action is:

RUN <n>

where n is a 4-digit number of instructions.

GO — Allows execution until either an error condition arises or a breakpoint is reached.

IGN — Allows the reported error to be ignored when the task is armed or when the task receives an asynchronous message.

QUIT Detaches the task from the session and ends the session. The task continues running.

NOTE

You should use QUIT to terminate a session where a system Independent Runner is being debugged; QUIT ensures that the task will be detached.

END Terminates both the task and the Debug session.

NOTE

You should not use END to end the session when debugging a system Independent Runner; you should use the QUIT action.

PEEK Displays memory at the indicated offset and memory area for the specified length.

The syntax for the PEEK action is:

```
PEEK <offset>, <length>, <memory area>,
     <environment number>, <task number>
```

The environment and task numbers are optional. The default values are the debugged task's current environment and task numbers.The digit lengths of these parameters are:

offset	— 6 digits
length	— 3 digits
memory area	— 2 digits
environment number	— 6 digits
task number	— 4 digits

POKE Allows you to change memory by inserting a "P" at the beginning of any lines in the display area.

RLE Displays the Reinstate List Entry of the specified task number.

MODE Changes the Debug mode to any of the following modes:

USR— enables Debug for the user program portions of the task only.

MCP— enables Debug for the MCP portion of the task only.

SYS— enables Debug for both the MCP and USER portions of the task

REL— disables Debug completely until either the task terminates or an error is reached.

For those portions of the task where Debug is disabled, the task will run at the normal processor speed. All actions which resume execution of the task or terminate the task can only be entered when the debugged task is in a stopped state.

Judicious use of these modes can reduce debugging time significantly. For example, if you want to debug a user program, there is no need to fault after each instruction executed within the MCP code. Use of the USR Mode would prevent this.

HELP Displays a series of help menus.

TRACE Displays the Trace Menu (described later in this section).

STATE Displays the State Menu (described later in this section).

ODT Allows you to switch to the ODT. You must use the ED command (described earlier in this section) to return to the Debug session from the ODT.

BREAKPOINT Displays the Breakpoint Menu (described later in this section).

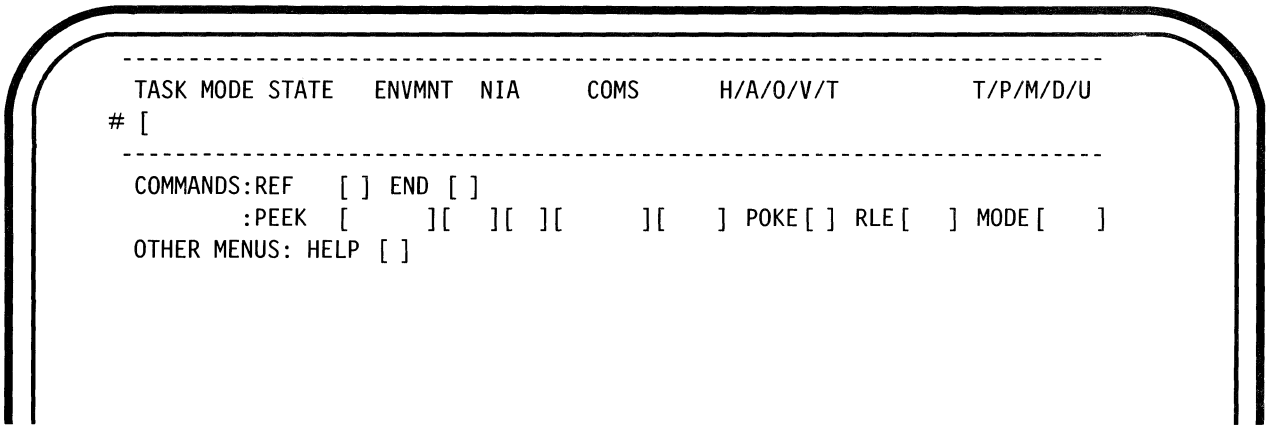


Figure 25-9. Status Menu

The Status Menu

The Status Menu (refer to Figure 25-9) appears when you invoke a DEBUG or ID command (described earlier in this section) when the task to be debugged is in a non-runnable state (that is, a waiting state, such as when the program is waiting on a resource).

When the task becomes runnable, the Main Menu (refer to Figure 25-8) appears.

Status Menu Actions

The following actions can be invoked on the Status Menu.

REF Refreshes the screen by redisplaying the current menu.

END Terminates both the task and the Debug session.

PEEK Displays memory at the indicated offset and memory area for the specified length.

The syntax for the PEEK action is:

PEEK <offset>, <length>, <memory area>,
<environment number>, <task number>

The environment and task numbers are optional. The default values are the debugged task's current environment and task numbers. The digit lengths of these parameters are:

offset — 6 digits
length — 3 digits
memory area — 2 digits
environment number — 6 digits
task number — 4 digits

POKE Allows you to change memory by inserting a "P" at the beginning of any lines in the display area.

RLE Displays the Reinstate List Entry of the specified task number.

MODE Changes the Debug mode to any of the following modes:

USR— enables Debug for the User Program portions of the task only.

MCP— enables Debug for the MCP portion of the task only.

SYS— enables Debug for both the MCP and USER portions of the task.

REL— disables Debug completely until either the task terminates or an error is reached.

For those portions of the task where Debug is disabled, the task will run at the normal processor speed. All actions which resume execution of the task and terminate the task can only be entered when the debugged task is in a stopped state.

Judicious use of these modes can reduce debugging time significantly. For example, if you want to debug a user program, there is no need to fault after each instruction executed within the MCP code. Use of the USR Mode would prevent this.

HELP Displays a series of help menus.

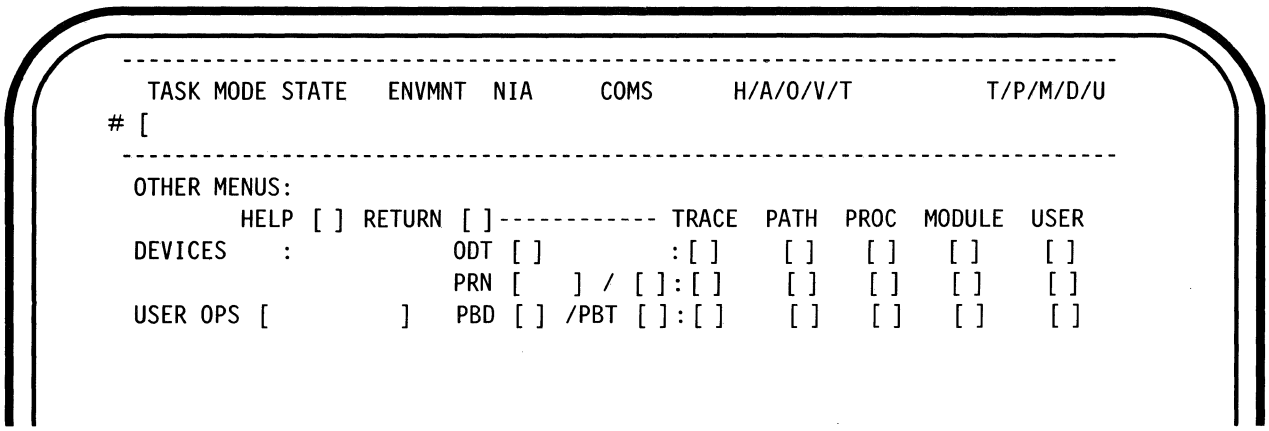

TASK MODE STATE ENVMNT NIA COMS H/A/O/V/T T/P/M/D/U
[
OTHER MENUS:
HELP [ ] RETURN [ ]
TRACE PATH PROC MODULE USER
DEVICES :
ODT [ ] :[ ] [ ] [ ] [ ] [ ]
PRN [ ] / [ ]:[ ] [ ] [ ] [ ] [ ]
USER OPS [ ]
PBD [ ] /PBT [ ]:[ ] [ ] [ ] [ ] [ ]

Figure 25-10. Trace Menu

The Trace Menu

You access the Trace Menu (refer to Figure 25-10) from the Main Menu (refer to Figure 25-8). This menu allows you to direct a variety of trace output to three types of media at the same time.

Trace Menu Actions

The following actions can be used on the Trace Menu:

HELP	Displays a series of help menus.
RETURN	Allows you to return to the Main Menu.

The types of trace actions are:

TRACE	This traces all instructions executed until the task terminates, an error occurs, or a breakpoint is reached.
PATH	This produces a trace line when instruction execution ceases to be sequential, as with Taken Branch instructions and Enter.
PROC	This produces output lines whenever the task enters or exits a procedure.
MODULE	This produces output lines whenever the task enters or exits a procedure and the environment changes.
USER	This produces output lines whenever the task executes a user-selected opcode.

Output (Device) Options

Any combination of the trace actions can be output to all devices. The output options are:

ODT	Enter any non-blank character.
PRINTER	Enter <channel number>/<unit number>. <channel number> is a 4-digit number and the <unit number> is a 2-digit number.
PBD or PBT	Enter any non-blank character for Printer Backup Disk (PBD) or Printer Backup Tape (PBT). Only one printer backup file can be opened at a time.

User Selected Opcodes

The USER OPS field allows you to specify a maximum of ten 2-hex-character opcodes.

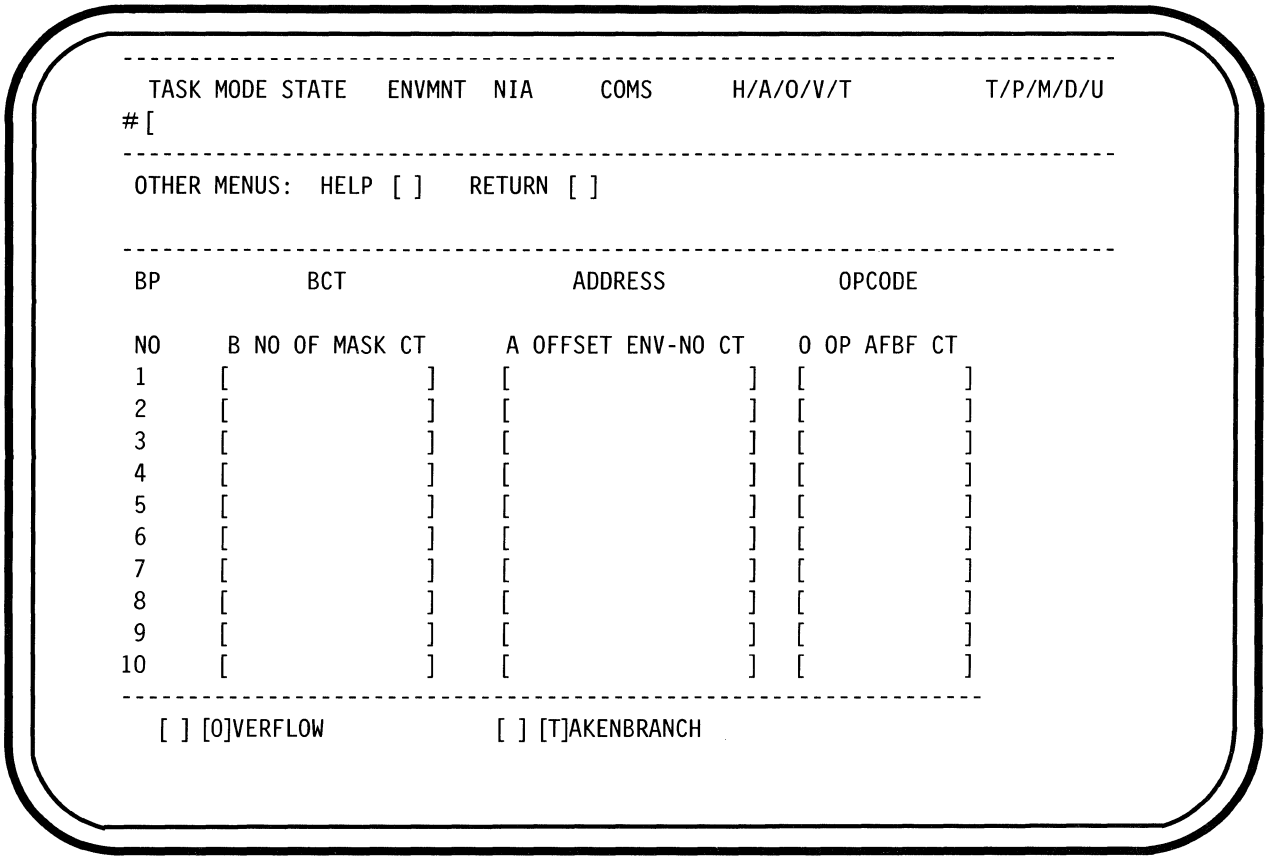

TASK MODE STATE ENVMNT NIA COMS H/A/O/V/T T/P/M/D/U
#[
OTHER MENUS: HELP [ ] RETURN [ ]
BP BCT ADDRESS OPCODE
NO B NO OF MASK CT A OFFSET ENV-NO CT O OP AFBF CT
1 2 3 4 5 6 7 8 9 10
[ ] [O]VERFLOW
[ ] [T]AKENBRANCH

Figure 25-11. Breakpoint Menu

The Breakpoint Menu

You access the Breakpoint Menu (refer to Figure 25-11) from the Main Menu (refer to Figure 25-8). This menu enables you to set a maximum of 32 breakpoints on 10 Hyper Call/BCT instructions, 10 instruction addresses, 10 opcodes, or an overflow or taken branch. All breakpoints can be easily set or reset by entering any non-blank character in the able/disable field which proceeds each breakpoint. The actions used for setting breakpoints are discussed below.

Breakpoint Menu Actions

You can use the following actions on the Breakpoint Menu:

HELP	Displays a series of help menus.
RET	Allows you to return to the Main Menu.

The Hyper Call/BCT Breakpoints

The actions for setting this breakpoint are shown in Table 25-1 with their default values.

Table 25-1. Hyper Call/BCT Breakpoints

Field	Description	Default
B	any character	not applicable
NO	4 digit number	0000
OF	2 digit number	00
MASK	6 hex value	F0F0F0
CT	2 digit number	00

Enter any character in the first field (B) to enable this breakpoint.

The second field (NO) assigns a four-digit number to this breakpoint. (For example, to stop on BCT 0214, you enter 0214. Refer to the examples later in this section.)

The offset (OF) is the number of digits relative to the start of the HCL / BCT parameters.

The mask (MASK) is compared with the offset value and the task is stopped on a perfect match.

The count (CT) specifies the number of occurrences of the Hyper Call / BCT to be skipped before the task is brought to a halt.

The Address Breakpoint

The actions necessary for setting this breakpoint are listed below with their default values.

Table 25-2. Address Breakpoints

Field	Description	Default
A	any character	not applicable
OFFSET	6 digit number	000000
ENV-NO	6 digit number	000000
CT	2 digit number	00

Enter any character in the first field (A) to enable this breakpoint.

The address offset (OFFSET) is the instruction address on which the task is to be stopped.

The Environment Table number (ENV-NO) is the MAT number.

The count (CT) specifies the number of occurrences of the address that are to be skipped before the task will stop.

The Opcode Breakpoint

This breakpoint stops execution at a given number of occurrences of a specified instruction. The actions for setting this breakpoint are listed below with their default values.

Table 25-3. Opcode Breakpoints

Field	Description	Default
O	any character	not applicable
OP	3 character string	3 blanks
AFBF	4 digit number	FFFF
CT	2 digit number	00

Enter any character in the first field (O) to enable this breakpoint.

The symbolic opcode (OP) is the mnemonic for the opcode.

The AFBF field is the actual value that is compared to the code stream before the task is stopped.

The count specifies the number of occurrences of the opcode ignored before the task is stopped.

The Overflow Breakpoint

You can stop the task once an executed instruction causes the Overflow Toggle to be set.

The Taken Branch Breakpoint

This breakpoint allows you to stop the task once the sequential flow of the instruction stream has been broken.

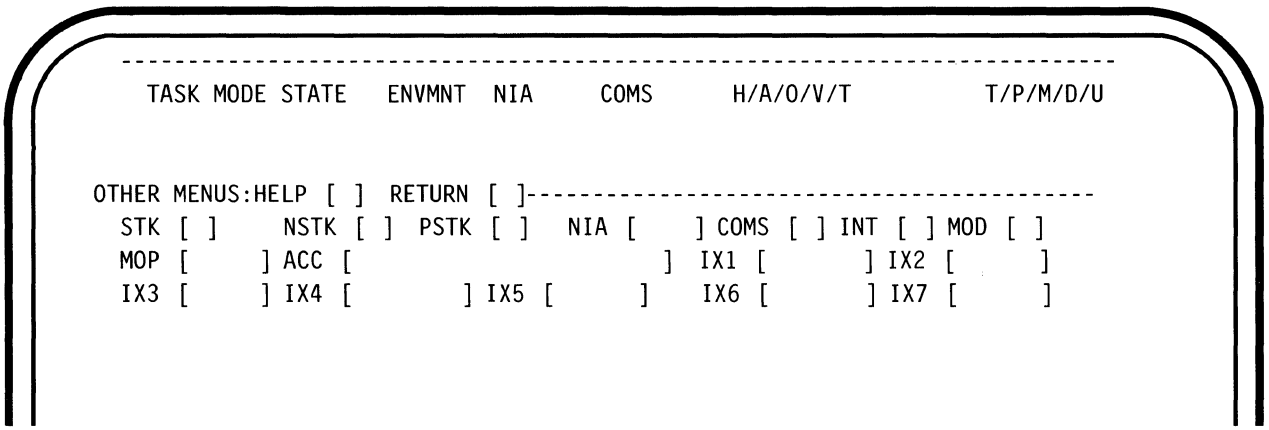

TASK MODE STATE ENVMNT NIA COMS H/A/O/V/T T/P/M/D/U
OTHER MENUS:HELP [ ] RETURN [ ]
STK [ ] NSTK [ ] PSTK [ ] NIA [ ] COMS [ ] INT [ ] MOD [ ]
MOP [ ] ACC [ ] IX1 [ ] IX2 [ ]
IX3 [ ] IX4 [ ] IX5 [ ] IX6 [ ] IX7 [ ]

Figure 25-12. State Menu

The State Menu

You can display and modify the task's state and data structure in memory by accessing the State Menu (refer to Figure 25-12) from the Main Menu (refer to Figure 25-8).

State Menu Actions

The following is a list of the actions for this menu and their functions.

HELP Displays a series of help menus.

RET Allows you to return to the Main Menu.

STK This displays the Topmost Stack Frame.

NSTK This displays the Next Stack Frame relative to the last one displayed.

PSTK This displays the Previous Stack Frame relative to the last one displayed.

NIA This displays/modifies the Next Instruction Address.

COMS This displays/modifies the task's Comparison Toggles.The encodings are as follows:

O - Overflow is set	COM 04
E - COMS are set	EQUALCOM 03
L - COMS are set	LOWCOM 02
H - COMS are set	HIGHCOM 01
N - COMS are set	NULLCOM 01

INT This displays/modifies the Interrupt Mask.The encodings are as follows:

O - Processor over temperature	INT 20
T - Timer	INT 10
R - Real-Time I/O	INT 04
E - Error I/O	INT 02
N - Normal I/O	INT 01
O - No bits set	INT 00

MOD This displays/modifies the Mode Toggles.The encodings are as follows:

F - Soft Fault enabled	MODE 08
P - Privileged Task	MODE 04
T - Trace enabled	MODE 02
S - Snap enabled	MODE 01
O - No toggles set	MODE 00

MOP This displays/modifies the Measurement Register.

ACC This displays/modifies the current Accumulator.

IXn This displays/modifies one of the processor's Index Registers, where "n" is a number between 1 and 7, inclusive.

DEBUG SESSION EXAMPLES

The following examples show the use of the Debug interactive menus.

Stop User Program

To stop a user program on an instruction address, you mark the Breakpoints fields on the Main Menu (refer to Figure 25-13).

When you transmit the Main Menu, the system displays the Breakpoint Menu (refer to Figure 25-14).

```
-------------------------------------------------------------------------------
  TASK MODE STATE   ENVMNT  NIA     COMS      H/A/O/V/T          T/P/M/D/U
#[ 002  SYS DEBUG   D00002  071930   02
-------------------------------------------------------------------------------

CMNDS:REF  [ ] STOP [ ] SI [ ] RUN [0000] GO[ ] IGN [  ] QUIT [ ] END [ ]

    :PEEK [000000][000][00][      ][    ] POKE[ ] RLE[000000] MODE [   ]
 OTHER MENUS: HELP [ ] TRACE [ ] BREAKPOINTS [X] STATE [ ] ODT [ ]

-------------------------------------------------------------------------------


                    TASK IS ATTACHED AND ENABLED
```

Figure 25-13. Main Menu—Breakpoint Example

```
-------------------------------------------------------------------------
  TASK MODE STATE   ENVMNT  NIA     COMS      H/A/O/V/T          T/P/M/D/U
#[ 002  SYS BRK-PNT D00002  071930   02
-------------------------------------------------------------------------
 OTHER MENUS:  HELP [ ]   RETURN [ ]

-------------------------------------------------------------------------
 BP           BCT                 ADDRESS              OPCODE

 NO     B NO OF MASK CT      A OFFSET ENV-NO CT    O OP AFBF CT
 1      [               ]    [X 005258 000001   ]  [            ]
 2      [               ]    [                  ]  [            ]
 3      [               ]    [                  ]  [            ]
 4      [               ]    [                  ]  [            ]
 5      [               ]    [                  ]  [            ]
 6      [               ]    [                  ]  [            ]
 7      [               ]    [                  ]  [            ]
 8      [               ]    [                  ]  [            ]
 9      [               ]    [                  ]  [            ]
10      [               ]    [                  ]  [            ]
-------------------------------------------------------------------
   [ ] [O ]VERFLOW             [ ] [T ]AKENBRANCH
```

Figure 25-14. Breakpoint Menu— Address Breakpoint Example

Transmit this menu and return to the Main Menu (use the RETURN field). The Main Menu is shown in Figure 25-15.

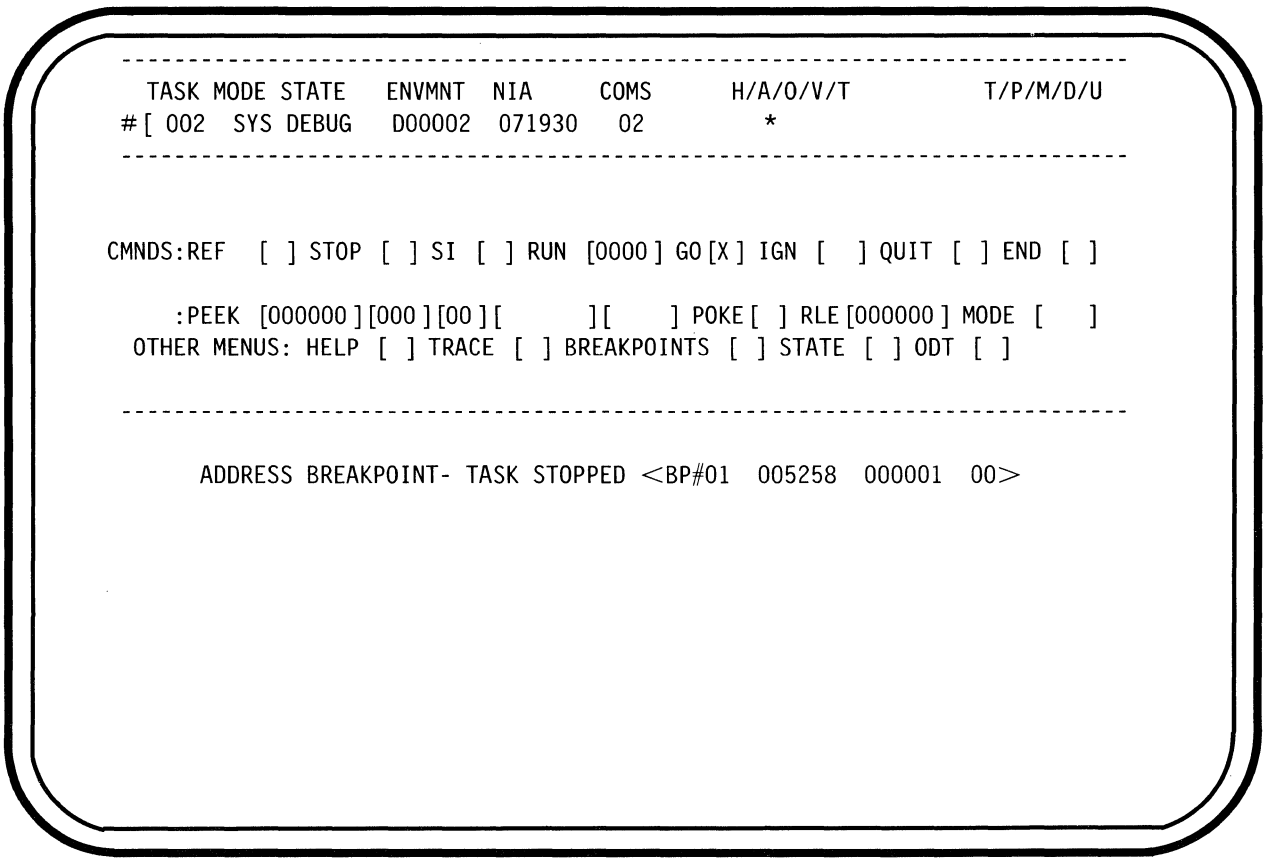

TASK MODE STATE ENVMNT NIA COMS H/A/O/V/T T/P/M/D/U
#[002 SYS DEBUG D00002 071930 02 *
CMNDS:REF [ ] STOP [ ] SI [ ] RUN [0000] GO[X] IGN [ ] QUIT [ ] END [ ]
:PEEK [000000][000][00][ ][ ] POKE[ ] RLE[000000] MODE [ ]
OTHER MENUS: HELP [ ] TRACE [ ] BREAKPOINTS [ ] STATE [ ] ODT [ ]
ADDRESS BREAKPOINT- TASK STOPPED <BP#01 005258 000001 00>

Figure 25-15. Main Menu— Example

Place a mark in the GO field and transmit the menu. The system executes the breakpoint and displays a message showing the breakpoint number, the offset number, the environment number, and the count number.

Using the PEEK and POKE Functions

The PEEK function displays the memory at the indicated address. The POKE function allows you to change the memory displayed.

The PEEK Function

On the Main Menu (refer to Figure 25-16), complete the first three fields of the PEEK field (the environment field and task number field are optional). Make sure that the GO field is blank.

```
---------------------------------------------------------------------------
  TASK MODE STATE   ENVMNT  NIA     COMS      H/A/O/V/T          T/P/M/D/U
#[ 002  SYS DEBUG   D00002  071930   02
---------------------------------------------------------------------------


CMNDS:REF  [ ] STOP [ ] SI [ ] RUN [0000] GO[ ] IGN [  ] QUIT [ ] END [ ]
       :PEEK [003292][012][01][      ][    ] POKE[ ] RLE[000000] MODE [   ]
 OTHER MENUS: HELP [ ] TRACE [ ] BREAKPOINTS [ ] STATE [ ] ODT [ ]

---------------------------------------------------------------------------
```

Figure 25-16. Main Menu— PEEK Input Example

The system displays the contents of the memory at location 003292 for a length of twleve digits (refer to Figure 25-17).

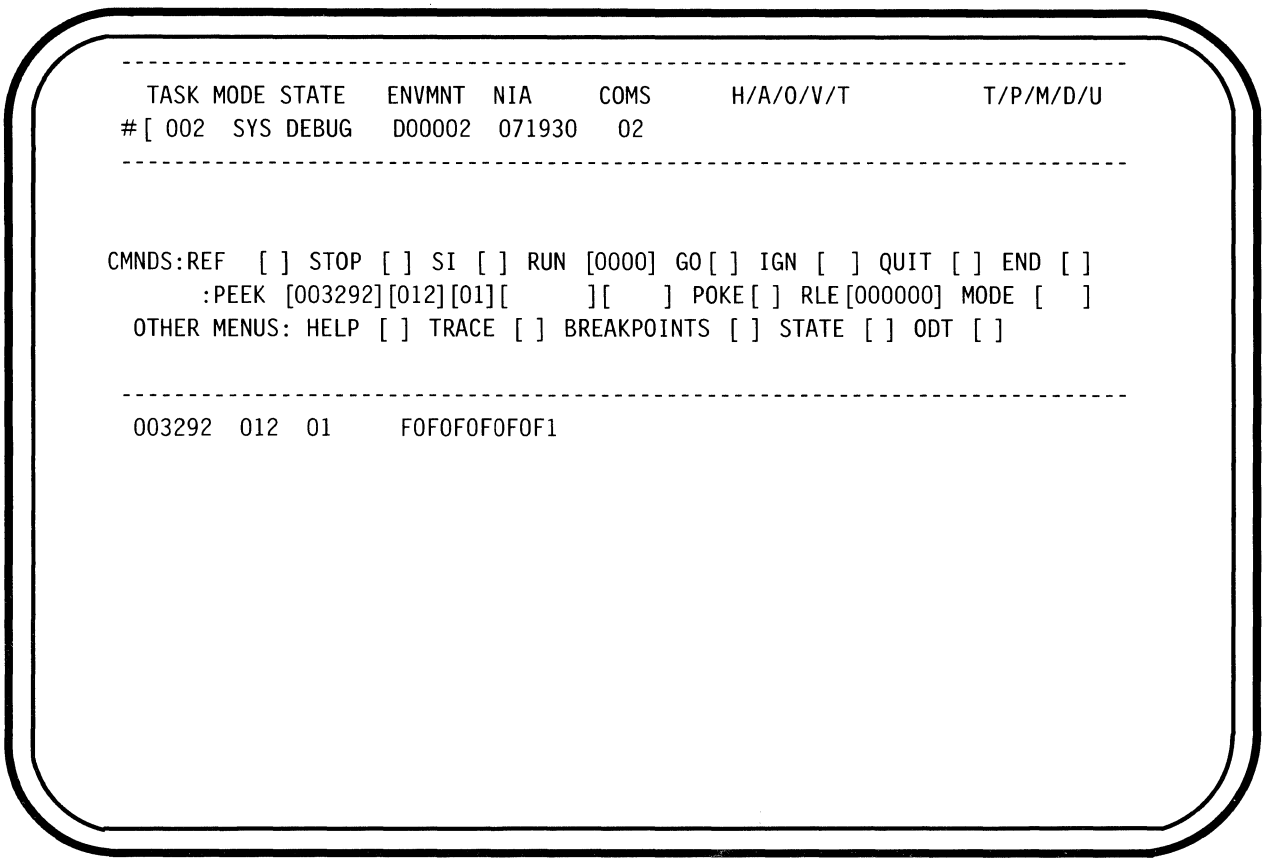

TASK MODE STATE ENVMNT NIA COMS H/A/O/V/T T/P/M/D/U
#[002 SYS DEBUG D00002 071930 02
CMNDS:REF [ ] STOP [ ] SI [ ] RUN [0000] GO[ ] IGN [ ] QUIT [ ] END [ ]
:PEEK [003292][012][01][ ][ ] POKE[ ] RLE[000000] MODE [ ]
OTHER MENUS: HELP [ ] TRACE [ ] BREAKPOINTS [ ] STATE [ ] ODT [ ]
003292 012 01 F0F0F0F0F0F1

Figure 25-17. Main Menu— PEEK Display Example

The POKE Function

In order to change the instructions in the memory location shown in Figure 25-17, you would use the POKE function.

Perform the following (refer to Figure 25-18):

1. Clear the PEEK fields.
2. Mark the POKE field.
3. Move the cursor to the display of the memory address.
4. Place a P immediately before the address number.
5. Change the memory contents.
6. Transmit.

```
------------------------------------------------------------------------
  TASK MODE STATE   ENVMNT  NIA     COMS      H/A/O/V/T          T/P/M/D/U
#[ 002  SYS DEBUG   D00002  071930   02
------------------------------------------------------------------------


CMNDS:REF  [ ] STOP [ ] SI [ ] RUN [0000] GO[ ] IGN [  ] QUIT [ ] END [ ]

    :PEEK [      ][   ][  ][      ][    ] POKE[X] RLE[000000] MODE [   ]
 OTHER MENUS: HELP [ ] TRACE [ ] BREAKPOINTS [ ] STATE [ ] ODT [ ]

------------------------------------------------------------------------



  P003292  012  01     F1F1F1F1F1F1
```

Figure 25-18. Main Menu— POKE Example

Using the Trace Function

The Trace Menu allows you to direct trace output to a variety of media.

Mark the Trace field on the Main Menu to access the Trace Menu (refer to Figure 25-19). Mark the desired fields. In this example, the request is for a complete trace of instructions to be sent to a printer backup file on disk (PBD).

When you transmit the menu, the system returns the message:

NEW PRINTER-BACKUP-FILE HAS BEEN OPENED

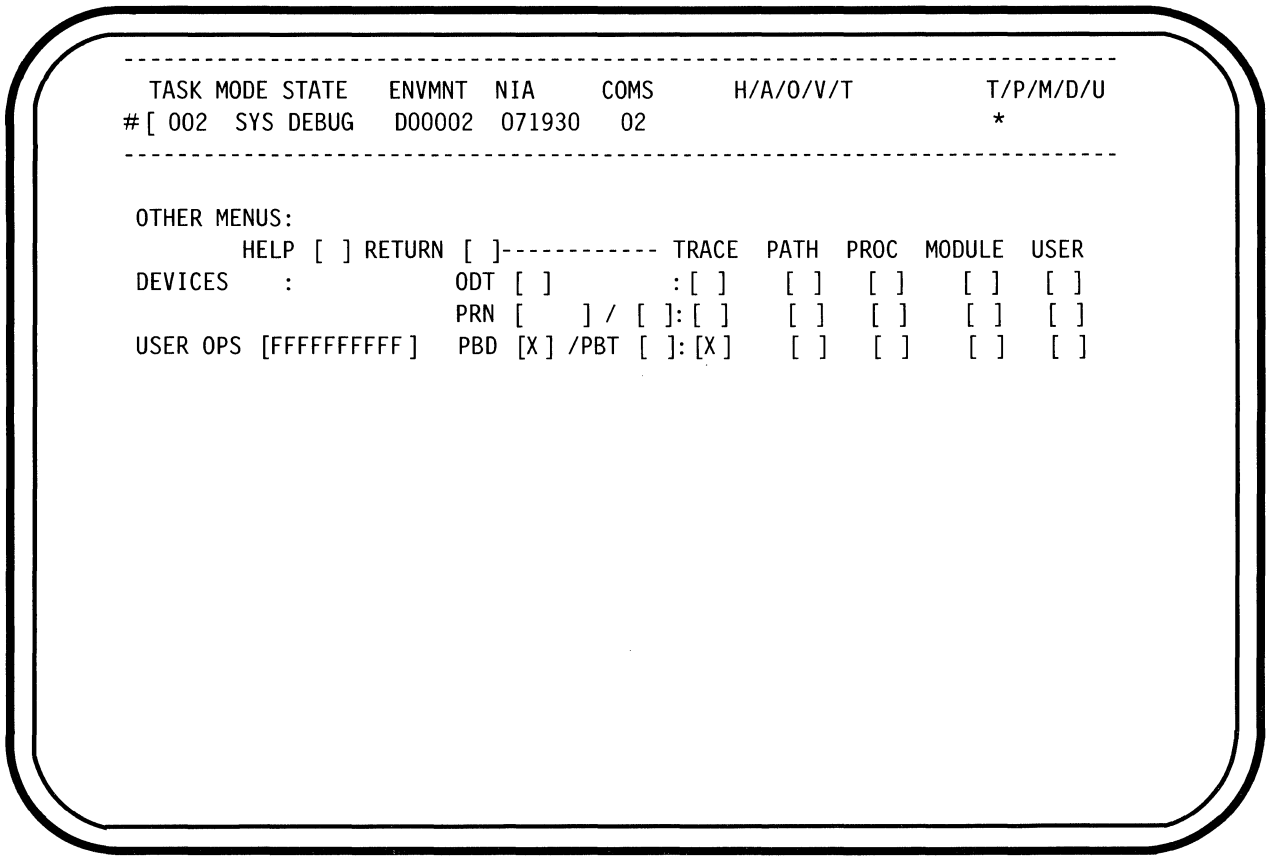

TASK MODE STATE ENVMNT NIA COMS H/A/O/V/T T/P/M/D/U
#[002 SYS DEBUG D00002 071930 02 *
OTHER MENUS:
HELP [ ] RETURN [ ]---------- TRACE PATH PROC MODULE USER
DEVICES : ODT [ ] :[ ] [ ] [ ] [ ] [ ]
PRN [ ] / [ ]:[ ] [ ] [ ] [ ] [ ]
USER OPS [FFFFFFFFFF] PBD [X] /PBT [ ]:[X] [ ] [ ] [ ] [ ]

Figure 25-19. Trace Menu— Example.

DEBUG SESSION ERRORS

When the processor is executing on behalf of a debugged task and encounters an error, it reports the error to the session through a Hardware Interrupt. The exact nature of the fault is indicated in the Fault Indicators.

Table 25-4. Fault Indicators

DIGIT	BIT	TYPE OF FAULT
72	8	Hard Memory Area
72	4	Trace
72	2	Invalid Arithmetic Data
72	1	Soft Memory Area
73	8	Invalid Instruction - More information in the IEX Byte; refer to Table 25-5.
73	4	Uncorrectable Memory Parity Error
73	2	Address Error - More information in the AEX Byte; refer to Table 25-6.
73	1	Instruction Timeout
74	8	Stack Overflow
74	4	Accumulator Trap taken
74	2	Snap Picture taken
74	1	Soft Fault

Table 25-5 lists the additional information that the invalid command extensions (digits 80-81) generate.

Table 25-5. Invalid Command Extension (IEX)— Digits 80-81

Description	Value
GENERAL	00
Invalid Operator Code	01
Privileged Mode Violation	02
Invalid Address Controller	03
Stack Overflow	04
Counter Overflow/Underflow	05
Invalid Field Comparison	06
Invalid Operand Field	07
Invalid AF or BF, GENERAL	20
Literal Not Allowed	21
Invalid Literal	22
Invalid IndirectField Length	23
Invalid Variant	24
Invalid AF	25
Invalid BF	26
Invalid Privileged Primary Access	31
Invalid Privileged Sec. Access	32
Invalid Attempt to Modify Original or Fault Memory Table Entry	35
Copy Protection Violation	36
Stack Protection Violation	37

Table 25-6 lists the additional information that the invalid address extensions (digits 78-79) generate.

Table 25-6. Invalid Command Extension (IEX)— Digits 78-79

Description	Value
GENERAL	00
Invalid Address Relationship	01
Hyper Call Function Limit Error	02
Odd Operand Address	03
Invalid MAT Entry	04
Index Register, GENERAL	10
Invalid Arithmetic	11
Index Register Contains Undigit	12
Invalid Base Indicant	13
IX3 Negative on RETURN	14
IX3 Odd on RETURN	15
Base/Limit Error, GENERAL	20
Command Fetch	21
Address Resolution	22
Operand Write	23
Operand Read	24
Global Link Address	25
Address Undigit, GENERAL	30
Command Fetch	31
Address Resolution	32
Operand Write	33
Operand Read	34
Global Link Address	35
Branch Address, GENERAL	40
Address >= Limit	41
Address Contains Undigit	42
Odd Address	43
Invalid Environment Descriptor	50
Invalid Environment Number	51
Invalid Most Significant Digit	52
Index Contains Undigit	53
Memory Area No. Contains Undigit	54
Environment Number or Memory Area Number Out of Range, GENERAL	56
Environment No. Out of Range	57
Memory Area No. Out of Range	58
Invalid Memory Area Table Entry	60
Invalid Environment Number	61
Invalid Most Significant Digit	62
Index Contains Undigit	63
Memory Area No. Contains Undigit	64
Environment Number or Memory Area Number Out of Range, GENERAL	66
Environment No. Out of Range	67
Memory Area No. Out of Range	68

ERROR MESSAGES RELATED TO DEBUG COMMANDS

The error messages which the DEBUG, ID, and ED commands can generate are listed below, along with their meanings.

EXCEEDED DEBUG SESSION LIMIT

There are currently ten active Debug sessions on the system. No new sessions can be invoked until at least one of the current sessions terminates. You can use the QD command to identify the current sessions.

ERROR IN INITIATING DEBUG SESSION

There has been an internal error in the sessions.

TASK CURRENTLY BELONGS TO ANOTHER SESSION

The task that you are attempting to attach to this session is currently being debugged in another session.

DEBUG SESSION DOES NOT BELONG TO OCS

The session number does not correspond with the ODT on which this command was issued.

DEBUG SESSION UNASSIGNED

The session number is invalid and does correspond to any current session.

FOUR DIGIT SESSION NO REQD

You must include a four-digit session number in the command.

SECTION 26
MEMORY DUMP FILE COPY UTILITY (DMPCPY)

The MCP/VS operating system creates a memory dump file after a 0 DM command is entered or when the system fails. The memory dump file can be stored on disk, diskpack or magnetic tape. Because of its large size, many installations choose to store the file on tape. For information about how a memory dump file is directed to tape, refer to the USE DUMP record in the *V Series MCP/VS System Software Operations Guide, Volume 1: Installation* or the PM command in the *V Series MCP/VS System Software Operations Guide, Volume 2: System Commands.*

After the memory dump file is stored on tape, normal tape-handling utilities (for example SYSTEM/COPY or DMPALL) cannot copy the file onto other tapes. This is because the format of a memory dump file is not compatible with the normal tape utility programs. To copy a memory dump file from one tape to another, use the utility program DMPCPY.

EXECUTING DMPCPY

Mount the tape containing the memory dump file on a tape drive. Use the RY command to ready the drive.

Enter the system command

EX DMPCPY

No special syntax is needed to execute DMPCPY. DMPCPY automatically searches for an input tape file named $00001. If no such file exists on tape, the program is marked as waiting and waits until a tape with a file named $00001 is available.

When the program finds the input file, it requests a scratch tape to use for the output file. If the end of the output tape is encountered before the end of the input tape, a message is displayed on the ODT asking the operator to mount another tape.

When an I/O error is encountered, the message

ERROR ON INPUT TAPE. <MIX> AX TO CONTINUE

is displayed on the ODT. In almost all situations, it is important to have an exact copy of the memory dump file. For this reason, it is better to execute DMPCPY again with another tape or a different drive, rather than continue with errors.

SECTION 27
OBJECT PROGRAM COPY UTILITY (PCOPY)

GENERAL

PCOPY is a bound system utility program that copies and modifies program code files or WFL code files. PCOPY transfers code files from 100-byte disk to 180-byte diskpack and vice versa. Thus, all code files except user-written system intrinsics such as PBDOUT, RLGOUT, and time-sharing programs may reside on diskpack and need not be reloaded with each cold start. The diskpack or diskpack family on which the programs reside may be either shared or non-shared. Refer to the CONTROL CODEPACK record in the *V Series System Software Operations Guide, Volume 1: Installation* for more information about code files on diskpack.

OPERATING INSTRUCTIONS

PCOPY is executed with the following syntax:

EX PCOPY (100 , "<program-ID>" , "<diskpack-ID>")

where 100 is an optional parameter setting VALUE 0 equal to 1.

The specified <program-ID> must be resident on 100-byte disk or 180-byte diskpack and must be a valid code file. Self-explanatory error messages are produced if these criteria are not met.

- If the code file is transferred from 100-byte disk to 180-byte diskpack, the <diskpack-ID> must specify the the diskpack or diskpack family where the code file is to reside. It must be a valid full diskpack name or family name (not a diskpack-group-ID). Transferring from 100-byte disk to 180-byte diskpack is the default PCOPY direction.
- To transfer the code file from 180-byte diskpack to 100-byte disk, the VA 0 1 option must be used. By using this option, you indicate that the destination of the code file is 100-byte disk, and that <diskpack-ID> specifies the name of the diskpack or diskpack family where the code file currently resides.

Once the code file is transferred to diskpack, it may be executed in the normal manner. (See EXECUTE and COMPILE statement syntax).

NOTE

PCOPY is not necessarily needed when using the COBOL ANSI-74 and RPG compilers (Release ASR 6.5, and greater). These compilers are capable of generating code files directly to pack in the proper format. Refer to COMPILE statement syntax in V Series MCP/VS System Software Operations Guide, Volume 1 for proper syntax.

Examples:

EX PCOPY/COBOL/SPLIBS

This transfers the COBOL compiler to the diskpack family named SPLIBS. The compiler can then be used to compile a program with a statement of:

CMP JAKE WITH COBOL ON SPLIBS LIB.

This creates a new program called JAKE on DISK.

EX PCOPY/COBOL/SPLIBS VA 0 1

This transfers the code file named COBOL from the diskpack family named SPLIBS to 100-byte disk.

SECTION 28
TRANSLATION FILE GENERATOR PROGRAM (MAKTRN)

MAKTRN creates a disk file used with the key translation option of the SORT intrinsic utility programs (SORT. and SORT:). This option is also known as the COLLATING SEQUENCE option in COBOL. MAKTRN creates a 400-byte, single-record, single-area file on 100-byte disk. The file contains a translate table in the format expected by the SORT intrinsic programs.

MAKTRN reads parameter records that describe either an entire collating sequence or only the non-standard portions of a collating sequence. The parameter are records checked for correct syntax. If no syntax errors are found, MAKTRN creates a valid translate table.

Using MAKTRN, you can:

- Create a complete new collating sequence for sorting.
- Interchange specific characters, retaining the remainder of the normal collating sequence.
- Assign the same collated value to multiple characters.

OPERATING INSTRUCTIONS

Use the following pseudo deck or control cards to execute MAKTRN:

```
?EXECUTE MAKTRN
?DATA CARD

  <parameter records>

?END
```

MAKTRN PARAMETER RECORDS

The format of <parameter records> is as follows.

- All parameter records have a dollar sign ($) in column 1.
- A valid MAKTRN option name must appear in columns 2-5.
- Columns 6-71 are free form. You cannot continue characters from one record to another record.
- Columns 72-80 are not used by MAKTRN.

MAKTRN OPTIONS

The valid MAKTRN options are IDNT, ALFA, NUMR, and SEQN.

IDNT Option

IDNT determines the file name of the translate file being created. IDNT is not required. If you omit this option, MAKTRN assigns a default name of TRANS*. When IDNT is used, it must be the first option in the parameter records.

Example:

```
$IDNT TRANS1
```

This example assigns the name TRANS1 to the translation file created by MAKTRN.

ALFA Option

ALFA indicates changes in a collating sequence. To translate a character to a different character, include the old and new characters next to each other, without intervening spaces. At least one space must appear between pairs of characters. The ALFA option is a convenient method of converting graphic characters.

Characters not included on ALFA (or NUMR) option records retain their standard position in the collating sequence. A character can appear as the first of a pair only once. As the second of a pair it can appear as many times as needed. This allows any number of characters to be collated to the same character.

Example:

```
$ALFA  ([  )]  -/
```

This example collates parentheses as brackets and dashes as slashes.

NUMR Option

The NUMR option uses two-digit numeric values to represent the old and new characters. NUMR can be used for any characters, but it is most convenient for non-graphic characters. As with the ALFA option, enter old and new characters in contiguous pairs. Separate the pairs with spaces.

Characters not included on NUMR (or ALFA) option records retain their standard position in the collating sequence. A character can appear as the first part of a pair only once. A character can appear as the second part of a pair as many times as needed. This allows any number of characters to be collated to the same character.

Example:

```
$NUMR 0040 81C1 82C2 83C3
```

This example collates NULL characters (00) as spaces (40) and lowercase letters a, b, and c as uppercase A, B, and C.

SEQN

SEQN directs MAKTRN to create an entire new collating sequence in the translate file. If SEQN appears in the parameter records, ALFA and NUMR are not valid.

SEQN requires that all 256 possible character combinations be included on the record. After entering the keyword $SEQN, enter the character that collates as 00, then the character for 01, and so on. A hyphen between characters denotes a range of values. Separate ranges of characters or single characters with spaces.

Example:

$SEQN 40-EF F0-FF 00-3F

This example creates a collating sequence of spaces, special characters, letters, numbers, and data communication control characters.

INDEX

$00001 22-1
100-byte disk subsystem 23-1
12-punch 5-5
180-byte format 10-1
5.25- or 8-inch floppy disks 11-1
8-bit signed numeric 5-8
A
A Series printer backup tape 5-20
ABANDONED 2-8
algebraic operators 5-7, 5-8
alphanumeric format 5-7
alphanumeric or hexadecimal literal 19-9
alternate I/O path 7-4
ANSI label 5-15
ANSI-68 COBOL 17-1
APCR = 0 8-5
APCR = 1 8-5
APCR facility 8-5
ASCII 18-3
ASCII files 5-2
ASCII/EBCDIC hardware strap 21-6
asynchronous 21-6
automatically direct printer backup files to the Image Printer 14-6
available table 15-1
B
B974LD 1-1
B974MCS trace 1-1
bad sector 15-3
bad sectors 10-1, 15-1
bad tracks 11-14
BASE CLEAR 16-10
base diskpacks 7-5
basic label area 2-11
BAUD RATE 21-3
BAUD rates for the Uniline type 2B 21-6
BAUD rates for the Uniline types 2, 2A, and 2C 21-6
BCL 17-1, 18-3
BIN 5-10
Binary Interlaced in 100 or 180-byte mode 7-7
Binary Sequential in 100 or 180-byte mode 7-7
binary to binary 5-10
Block Lockout Table (BLT) 16-1
BLOCK LOCKOUT TABLE (BLT) 16-9
blocking-factor 5-1
BLT contention 16-8
bottleneck 23-15
BPL 17-1
B 874 firmware files 13-1
B 974 13-1
B 974 NDL source code 1-1
BUFFER 3-2
BUFFERSIZE 2-3
C
cache 23-11, 23-15
card deck 8-1
card images 8-5
carriage control tape 5-5
channel 23-14
COBOL compiler 23-19
code file overlays 23-30
coldstart 20-1
collating sequence 17-2
column number 14-3
Compilation rates 23-19
COMPILE and GO 8-6
concurrently access files 16-1
console DLP 4-3
contention cycle 16-8
CONTINUE 2-8
CONTROL CARD ERROR 8-4
control deck 8-1
control file 18-5
converting a file 5-14
COPY 2-1
copy files to and from the floppy disk 11-4
COPYADD 2-1
Copying an entire floppy disk to another floppy disk 11-12
Copying files from a floppy disk to disk and diskpack 11-12
copying files to and from a floppy disk 11-2
Copying image files to and from floppy disk 11-12
corrupt data 15-3
cover page of a printout 14-2
CR code 21-6
CRUNCH 5-19
CV320A 10-1
cylinder 7-33
cylinders 7-1
D
DASDL format 19-2
data base files 23-12
data blocks 2-12
data communications equipment 21-1
data communications parameters 21-1
Data Communications Processor (DCP) 3-1, 9-1, 13-1
data files 23-12

INDEX (Cont)

data integrity flag 2-8
data terminal equipment (DTE) 21-1
Decimal Interlaced in 100 or 180-byte mode 7-7
Decimal Sequential in 100 or 180-byte mode 7-7
DIR 2-1
directory listing of a library tape produced by the LOADMP or PACKUP programs 5-21
disk based system 23-11
disk channels 23-14
disk controller 10-2
disk sharing is determined at the subsystem level 16-10
diskpack (180-byte sectors) 7-1
diskpack based system 23-11
diskpack controller 9-1
diskpack controllers 10-1
diskpack controllers - Bx383, Bx384, Bx385, B9387 10-1
diskpack family name 7-5
diskpack in 100-byte format 10-1
diskpack initialization functions 7-1
diskpack preparation in 100-byte mode 7-36
Diskpack serial number can range from 1 to 999999 15-2
diskpack serial numbers 15-1
diskpack sharing is determined at the family level 16-10
diskpack type and access restriction 7-5
DLP 4-1
DLPXNO and DLPXCO 4-1
DMPANL 20-1
DMPANL program reads an MCP memory dump from disk, diskpack, or tape 22-1
DMPMEM 6-1
DMSII 23-3
DMSII index files (sets and subsets) are good candidates for QWIK Disk 23-3
dollar sign cards 17-4
double-spaced 5-5
DTE 21-3
dump 20-1, 22-1
dump analysis 6-1
dump and clear data link processors (DLPs) 4-1
dump file from disk or diskpack to tape 22-2
dump file ($p0001) 20-2
dump, including the current task information, the function path, and the current task MCP stack frame
dump information 4-1
dump MCP memory 20-1
dump memory to tape 22-1
dump the MCP to tape 6-1
dumps 6-1
duplicate all the data on one diskpack on another diskpack 15-1
duplicate diskpack 15-1
E
EBCDIC 5-2, 8-4, 17-1, 18-3, 21-6
?ENDCTL 8-4
EOF detection 8-10
EOF (End-of-File) condition 8-10
ET 21-2
ETX code (03 hex) 21-6
EU# 17-5
Extended Storage Module Device 7-9
Extended Storage Module Device (XSMD) 7-7
F
family name 7-6
Fault Handler 6-1
FIBMBS 17-3
FIBMRL 17-3
FIBVAR 17-3, 17-4
FID 2-10
file called DCxxpP 9-1
file description block (FID) 2-12
File Information Block (FIB) 8-10
file locks held by a system 16-10
file selection criteria 23-17
FILE statements 19-2
file transfer 2-1
files or execution decks on floppy disk 11-3
firmware 9-1, 21-1
firmware code file 1-1
firmware file 9-1
firmware file blocked 100 bytes by 1 21-11
Firmware File USP2BH 21-2
Firmware File UST2BH 21-3
firmware level SSP302 16-10
flag digit 8-9
FLAME peripheral activity graphs 23-14
FLAME reports help summarize QWIK Disk overall performance 23-15
floppy disk directory 11-7
floppy disk drive units 0 and 1 11-1
floppy disk file conversion function 11-11
floppy disk formatting (initializing) 11-8
floppy disk initialization 11-9
floppy disk maintenance functions 11-1
floppy disks 11-1
formats a floppy disk 11-8
forms 14-3
Forms Manager 14-1

INDEX (Cont)

Forms Manager Program 14-3
forms printer 5-9
G
GEMCOS 3-1
Global Information Count 13-5
H
haltload 20-1, 23-26
haltload while pseudo card files are active 8-6
hardware collating sequence 18-6
hardware exceptions 23-30
hexadecimal characters 5-7
HOSTNAME 2-3
HOTSORT 19-7
I
IBMIRD 11-8
ICMD-DLP 11-1
ICTAPE format 2-10
Image Page Printer 14-3
Image Page Printer (IPP) 9-2
independent runners 22-5
indexed sequential 19-1
indexed sequential input file 19-4
Industry Compatible Minidisk (ICMD) 11-1
Initialization of a single system shared system 16-10
initialization of multiple shared systems 16-11
initialize/verify/relocate (IVR) functions 10-1
interchangeable media 2-10
interlace code 11-9
interlace code for a floppy disk 11-9
Interlaced in 100 or 180-byte mode 7-7
intersystem mailbox 16-10
invalid character 8-9
I/O 23-1
I/O trace of console DLP operations 4-1
I/O wait time 23-18
I/O wait time on individual files 23-12
I/O waiting time 23-3
IOAT entry 8-10
IOATs 22-5
IOP error 10-5
I/Os to disk or diskpack 23-10
irrecoverable I/O operations 4-1
ISAM 23-3
ISAM files 23-12
IVR process 7-11
K
key record fields 12-1
L
label 14-2
LAK diskpacks 10-1
latency time 23-3
LDCNTL 8-1
LDHOST-MCP Interfaces 9-2
LH 1-1
LH WARM 1-1, 1-2
library maintenance 2-5
link field value 8-9
LOAD MC 20-1
Loadable Operating System (LOS) 9-2
LOADFW 10-1, 21-1
LOADMP 2-10
LOCK NO CONTEND shared file I/O operation 16-8
LOCK READ shared file I/O operation 16-8
LOCK READ UNLOCK shared file I/O operation 16-8
LOCK SEEK shared file I/O operation 16-8
LOCK SEEK UNLOCK shared file I/O operation 16-8
LOCK shared file I/O operation 16-8
lock/unlock operations on a record level 16-2
logical subsystem 16-10
M
Magnetic tape (NRZ, phase-encoded, or GCR recording mode) 5-14
mailbox 16-8
maintenance logs 1-1
maintenance panel command 10-2
maintenance processor commands 23-23
Master 7-5
Master Clear is an electronic signal that the system uses to initialize a DLP 16-10
master diskpack 7-5, 7-6
MCP code file and shared the disk directory and available table 16-7
MCP disk must be shared 16-7
MCP dump 6-1
MCP idle time 23-4
MCP MID 22-2
MCP overlays 23-8
MCP system configuration file for shared operations 16-11
MCP tables 22-3, 22-5
MCPNIF 3-1
MCPNIF files 13-1
MCPNpF and MCSNpF 3-1
MCP/USER graphs produced by the FLAME software package can help you determine the MCP idle time 23-6
MCS 1-2, 3-1
MCSNIF files 13-1

INDEX (Cont)

memory 9-1, 12-1, 17-1, 17-2
Memory Area Status Table 22-7
MEMORY statement 19-9
memory utilization 23-4
MEMORY/DUMP 22-1
MERG: 12-1
MERGE 2-1
merge 18-4
merge intrinsic program 12-1
merge keys 12-1
MERGED MCPNIF 13-2
message routing 3-1
microcode address 4-1
microcode word 4-1
Micro-Minidisk Unit (MMDU) format 11-1
MID 22-2
missing file conditions 2-9
MMDU format floppy disks 11-1
Module Interface Description (MID) 22-2
MOVE 2-1
MT983 21-2
MTP 5-2
multi-file tape 5-1
multi-file-ID 2-13
multi-file-pack-ids 17-5
multi-system shared configuration 16-5
multisystem shared configuration 16-7
multi-system shared environment 16-4
N
Network Definition Language (NDL) 3-1
NIFMRG utility 13-1
non-contiguous areas 5-10
O
object code disk files in hexadecimal format 5-1
OCSs 21-1
ODT log 22-8
ON LINE switch 10-1
operating environment 20-1
ORS DLP 13-1
overlay performance 23-4
overlays 23-3
P
PACKUP 2-10
parity 5-1
parity error 2-7, 5-24
parity generation and checking 21-7
PARITY statement 19-9
pass file 2-1
pass files 22-2
PCF 14-2
PCRXRF 8-5
physical subsystem 16-10
PKCOPY 15-1
POLL operation 21-6
poll/select 21-3
Port Attribute Structure 22-7
Port Global Data Memory Area 22-7
pound sign 8-4
PR STOPPED 2-1
presorted input files 12-1
print all printer backup files 14-2
print automatically 14-6
print printer backup files from tape 14-2
printer backup disk 5-9
printer backup diskpack 5-9
printer backup file on disk 14-4
printer backup file on diskpack 14-4
printer backup files 14-1
printer backup listings 1-1
printer backup tape 5-1, 5-9
printer backup tapes 14-1
printers 20-1
printing 14-1
processor number 20-1
processor usage 23-4
production environment 23-15
programmable controller 9-1
pseudo card file 8-1
pseudo card file can contain one or more sets of control instructions or data files 8-7
pseudo card files 8-5
Pseudo-Reader Cross Reference Directory (PCRXRF) 8-5
Pseudo-Reader Directory 8-6
PUBLIC/IO 2-2
punched card file 5-6
Q
question mark 8-9
QWIK Disk 23-1
QWIK Disk ID 40/0 23-19
QWIKPOOL 23-3
QWKMEM 23-8
R
RAM memory 23-18
random or sequential I/O 23-30
READ I/O operation 16-8
RECIRCULATING POLL I/O OPERATION 21-7

INDEX (Cont)

Record Information Field (RIF) 2-12
Record Lockout Table 16-8
relative 19-1
reliability information 2-8
relocated sectors 7-1
RELOCATION 7-10
relocation of a sector 7-32
remote job entry terminals 21-1
remote terminal 5-22
Resident Operating System mode (ROS) 9-2
Restricted 7-5
Restricted Master 7-5
result descriptor (R/D) 10-5
Result Descriptors 23-30
RIF 2-12
RLGOUT reports detail file sizes and I/O activity 23-15
RS232 asynchronous direct connection 21-2
RS232 equipment and modems 21-1
run log analysis 23-13
S
SAVE WITH LOCK 5-19
S-code file create by the PTL compiler on an A Series System 11-11
S-code files 11-11
S-DUMP 9-1
security 2-2
security user table 22-7
SEEK I/O operation 16-9
seek time 23-1
segment dictionary table 22-7
sequential 19-1
Sequential in 100 or 180-byte mode 7-7
SHARED 16-4
shared configuration rules 16-4
shared diskpack resources 16-5
shared file I/O operations 16-3, 16-4, 16-8
shared files 16-4
shared record locking/unlocking protocol 16-5
Shared Subport Memory Area (SSMA) 22-8
shared system 16-1, 23-25
shared system configurations 16-5
Shared system processor (SSP) 16-10
shared system processors (SSP) 10-1
Shared Systems Processor (SSP) 16-5
sign digit 19-5
signed numeric field 19-5
single bit memory errors 23-30
single cylinder IVR 7-33
single shared system 16-4
single-spaced 5-5
single-system shared configuration 16-5
SMD 7-7, 9-1
SMD DLP II 9-1
S-memory 9-1
S-Memory Dump Flag 9-2
SORT. 17-1
sort 17-1
SORT BCT 17-4
SORT can accommodate input and output to and from any pseudo deck, tape, disk or diskpack 18-2
SORT. intrinsic can accept magnetic tape variable length records as input 17-3
SORT key 18-1
sort keys 17-2
SORT: shows marginal improvement with work files on QWIK Disk 23-3
sort with translate 17-2
sorting speed 17-1
SORT.p (where p is the processor number) 17-1
source tape file expansion 2-1
SRTUTL 19-1
SRTUTL performance can improve based upon placement of input, output, and/or work files on QWIK Disk 23-3
stack 4-1
stacked deck 16-11
stalemate 16-9
stalemate use routine 16-8
status count 4-1
status of a floppy disk 11-7
stopbits 21-7
Storage Module Device 9-1
Storage Module Device (SMD) 7-9
Storage Module Device (SMD) 100-byte mode 7-7
strapped 7-36
Subport Attribute Structure 22-7
Subport Directory 22-7
SUMMARY 2-5
$SUPR 18-6
switched (dial) or leased lines 21-3
system analysis 23-5
system initialization 23-24
system interchange code 7-6
SYSUP facility 20-1
T
table breakout and a raw dump of the MCP and overlay area 22-5

INDEX (Cont)

tape 2-9, 5-20
tape directory 2-6
tape file that has variable record length 5-9
tape name 5-5
tape reel 2-6
TAPE/DISK SORT intrinsic 18-1
TC 4000 21-2
TD8 21-2, 21-8
TELCOM DLP 13-1
telephone system 21-3
teletypewriters 21-1
Time Sharing Processor Information Block and the Shared Area Table 22-8
time-sharing and handlers 20-1
TM represents a tape mark 2-10, 2-12
track and sector structures on the diskpack 10-1
TRAK buffer 22-8
translate all lower case letters to upper case letters 14-1, 14-4
translate table file 18-6
U
UC-DLP 11-1
unattended, on line environment 20-1
Uniline 9-1
Uniline DLP 10-1
Uniline DLPs 21-1
Universal Console DLP 4-3
unlabeled file 5-15
UNLOCK shared file I/O operation 16-9
unrestricted diskpack subsystem 7-5
unsigned alphanumeric 18-6
unsigned alphanumeric data 5-8
unsigned numeric data 5-8
USASI Standard Label 2-11
USE AUHL 20-1
USE DUMP 22-1
USE SYSUP 20-1
USP2BH 21-1
UST2BH 21-1
V
variable block size 5-9
variable number of records per block 5-9
VERIFICATION 7-10
virtual collating sequence 18-6
volume ID of a floppy disk (multifile name) 11-7
W
WO reports on overlay activity in the QWIK pool 23-15
$WORK DISK 18-6
work files 23-3, 23-12
$WORK PACK 18-6
WRITE I/O operation 16-9
X